Approximation Algorithms and Hardness of Integral Concurrent Flow *

Parinya Chalermsook†
Dept. of Computer Science
University of Chicago,
Chicago, IL 60615
parinya@cs.uchicago.edu

Julia Chuzhoy ‡
Toyota Technological Institute
Chicago, IL 60637
cjulia@ttic.edu

Alina Ene§
Dept. of Computer Science
University of Illinois, Urbana,
IL 61801
ene1@illinois.edu

Shi Li ¶
Center for Computational
Intractability
Dept. of Computer Science,
Princeton University
shili@cs.princeton.edu

ABSTRACT

We study an integral counterpart of the classical Maximum Concurrent Flow problem, that we call Integral Concurrent Flow (ICF). In the basic version of this problem (basic-ICF), we are given an undirected n-vertex graph G with edge capacities $c(e)$, a subset $\mathcal{T}$ of vertices called terminals, and a demand $D(t,t')$ for every pair (t,t') of the terminals. The goal is to find a maximum value λ, and a collection $\mathcal{P}$ of paths, such that every pair (t,t') of terminals is connected by $\lfloor \lambda \cdot D(t,t') \rfloor$ paths in $\mathcal{P}$, and the number of paths containing any edge e is at most $c(e)$. We show an algorithm that achieves a $\operatorname{poly}\log n$-approximation for basic-ICF, while violating the edge capacities by only a constant factor. We complement this result by proving that no efficient algorithm can achieve a factor α-approximation with congestion c for any values α, c satisfying $\alpha \cdot c = O(\log\log n/\log\log\log n)$, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\operatorname{poly}\log n})$.

We then turn to study the more general group version of the problem (group-ICF), in which we are given a collection $\{(S_1,T_1),\ldots,(S_k,T_k)\}$ of pairs of vertex subsets, and for each $1 \le i \le k$, a demand D_i is specified. The goal is to find a maximum value λ and a collection $\mathcal{P}$ of paths, such that for each i, at least $\lfloor \lambda \cdot D_i \rfloor$ paths connect the vertices of S_i to the vertices of T_i, while respecting the edge capacities. We show that for any $1 \le c \le O(\log\log n)$, no efficient algorithm can achieve a factor $O\left(n^{1/(2^{2c+3})}\right)$-approximation with congestion c for the problem, unless $\mathsf{NP} \subseteq \mathsf{DTIME}(n^{O(\log\log n)})$. On the other hand, we show an efficient randomized algorithm that finds a $\operatorname{poly}\log n$-approximate solution with a constant congestion, if we are guaranteed that the optimal solution contains at least $D \ge k \operatorname{poly}\log n$ paths connecting every pair (S_i,T_i).

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems, Routing and Layout*.

General Terms

Theory, Algorithms

Keywords

Integral Concurrent Flow

1. INTRODUCTION

Multicommodity flows are ubiquitous in computer science, and they are among the most basic and extensively studied combinatorial objects. Given an undirected n-vertex graph $G=(V,E)$ with capacities $c(e)>0$ on edges $e \in E$, and a collection $\{(s_1,t_1),\ldots,(s_k,t_k)\}$ of source-sink pairs, two standard objective functions for multicommodity flows are: Maximum Multicommodity Flow, where the goal is to maximize the total amount of flow routed between the source-sink pairs, and Maximum Concurrent Flow, where the goal is to maximize a value λ, such that λ flow units can be simultaneously sent between every pair (s_i,t_i).

Many applications require however that the routing of the demand pairs is *integral*, that is, the amount of flow sent on each flow-path is integral. The integral counterpart of Maximum Multicommodity Flow is the Edge Disjoint

†Supported in part by NSF grant CCF-0844872.
‡Supported in part by NSF CAREER grant CCF-0844872 and Sloan Research Fellowship.
§Supported in part by NSF grants CCF-0728782, CCF-1016684 and CCF-0844872. Part of this work was done while the author was visiting TTIC.
¶Supported by NSF awards MSPA-MCS 0528414, CCF 0832797, AF 0916218 and CCF-0844872. Part of this work was done while the author was visiting TTIC.
*A full version of this paper is available at the authors' web pages.

STOCŠ12, May 19–22, 2012, New York, New York, USA.

Paths problem (EDP), where the goal is to find a maximum-cardinality collection $\mathcal{P}$ of paths connecting the source-sink pairs with no congestion. It is a standard practice to define the EDP problem on graphs with unit edge capacities, so a congestion of any solution $\mathcal{P}$ is the maximum number of paths in $\mathcal{P}$ sharing an edge. EDP is a classical routing problem that has been studied extensively. Robertson and Seymour [26] have shown an efficient algorithm for EDP when the number k of the demand pairs is bounded by a constant, but the problem is NP-hard for general values of k [19]. The best currently known approximation algorithm, due to Chekuri, Khanna and Shepherd [11], achieves an $O(\sqrt{n})$-approximation. The problem is also known to be $\Omega(\log^{1/2-\epsilon} n)$-hard to approximate for any constant ϵ, unless NP $\subseteq$ ZPTIME($n^{\text{poly}\log n}$) [3, 2]. A standard technique for designing approximation algorithms for routing problems is to first compute a multi-commodity flow relaxation of the problem, where instead of connecting the demand pairs with integral paths, we are only required to send flow between them. Such a fractional solution can usually be computed using linear programming, and it is then rounded to obtain an integral solution to the routing problem. For the EDP problem, the corresponding flow relaxation is the Maximum Multicommodity Flow problem. However, the ratio of the Maximum Multicommodity Flow solution value to the EDP solution value can be as large as $\Omega(\sqrt{n})$ in some graphs [11]. Interestingly, when the value of the global minimum cut in G is $\Omega(\log^5 n)$, Rao and Zhou [24] have shown a poly $\log n$-approximation algorithm for EDP, by rounding the multicommodity flow relaxation.

Much better results are known if we slightly relax the problem requirements by allowing a small congestion. Andrews [1] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\mathsf{OPT}/\operatorname{poly}\log n)$ of the demand pairs with congestion $\operatorname{poly}(\log\log n)$, where OPT is the value of the optimal solution with no congestion for the given EDP instance, and Chuzhoy [12] has shown an efficient randomized algorithm that w.h.p. routes $\Omega(\mathsf{OPT}/\operatorname{poly}\log k)$ of the demand pairs with a constant congestion. In fact the number of demand pairs routed by the latter algorithm is within a poly $\log k$-factor of the Maximum Multicommodity Flow value.

Assume now that we are given an instance where every demand pair (s_i, t_i) can simultaneously send D flow units to each other with no congestion. The algorithm of [12] will then produce a collection $\mathcal{P}$ of $\Omega(Dk/\operatorname{poly}\log k)$ paths connecting the demand pairs, but it is possible that some pairs are connected by many paths, while some pairs have no paths connecting them. In some applications however, it is important to ensure that **every** demand pair is connected by many paths.

In this paper, we propose to study an integral counterpart of Maximum Concurrent Flow, called Integral Concurrent Flow (ICF). We study two versions of ICF. In the simpler basic version (basic-ICF), we are given an undirected n-vertex graph $G = (V, E)$ with non-negative capacities $c(e)$ on edges $e \in E$, a subset $\mathcal{T} \subseteq V$ of k vertices called terminals, and a set $\mathcal{D}$ of demands over the terminals, where for each pair $(t_i, t_j) \in \mathcal{T}$, a demand $D(t_i, t_j)$ is specified. The goal is to find a maximum value λ, and a collection $\mathcal{P}$ of paths, such that for each pair (t_i, t_j) of terminals, set $\mathcal{P}$ contains at least $\lfloor \lambda \cdot D(t_i, t_j) \rfloor$ paths connecting t_i to t_j, and for each edge $e \in E$, at most $c(e)$ paths in $\mathcal{P}$ contain e.

The second and the more general version of the ICF problem that we consider is the group version (group-ICF), in which we are given an undirected n-vertex graph $G = (V, E)$ with edge capacities $c(e) > 0$, and k pairs of vertex subsets $((S_1, T_1), \ldots, (S_k, T_k))$. For each pair (S_i, T_i), we are also given a demand D_i. The goal is to find a maximum value λ, and a collection $\mathcal{P}$ of paths, such that for each $1 \leq i \leq k$, there are at least $\lfloor \lambda \cdot D_i \rfloor$ paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{P}$, and every edge $e \in E$ belongs to at most $c(e)$ paths. It is easy to see that group-ICF generalizes both the basic-ICF and the EDP problems[1]. As in the EDP problem, we will sometimes relax the capacity constraints, and will instead only require that the maximum edge congestion - the ratio of the number of paths containing the edge to its capacity - is bounded. We say that a set $\mathcal{P}$ of paths is a solution of value λ and congestion η, iff for every $1 \leq i \leq k$, at least $\lfloor \lambda \cdot D_i \rfloor$ paths connect the vertices of S_i to the vertices of T_i, and every edge $e \in E$ participates in at most $\eta \cdot c(e)$ paths in $\mathcal{P}$. Throughout the paper, we denote by λ^* the value of the optimal solution to the ICF instance, when no congestion is allowed. We say that a solution $\mathcal{P}$ is an α-approximation with congestion η iff for each $1 \leq i \leq k$, at least $\lfloor \lambda^* \cdot D_i/\alpha \rfloor$ paths connect the vertices of S_i to the vertices of T_i, and the congestion due to paths in $\mathcal{P}$ is at most η.

Given a multicommodity flow F, we say that it is a *fractional* solution of value λ to the group-ICF instance, iff for each demand pair (S_i, T_i), at least $\lambda \cdot D_i$ flow units are sent from the vertices of S_i to the vertices of T_i, and the total amount of flow sent via any edge e is at most c_e. Throughout the paper, we denote by λ_{OPT} the value of the optimal fractional solution to the ICF problem instance. The value λ_{OPT} can be efficiently computed by solving an appropriate LP-relaxation of the problem, for both basic-ICF and group-ICF. Observe that for basic-ICF, this is equivalent to solving the Maximum Concurrent Flow problem.

In addition to designing approximation algorithms for the ICF problem, an interesting question is the relationship between the optimal fractional and the optimal integral solutions for ICF. For example, suppose we are given a multicommodity flow, where for each $1 \leq i \leq k$, the vertices of S_i send D flow units to the vertices of T_i simultaneously with no congestion. What is the maximum value λ, for which we can find an integral solution where for each pair (S_i, T_i) at least $\lfloor \lambda D \rfloor$ paths connect the vertices of S_i to the vertices of T_i?

We start by showing a randomized algorithm for basic-ICF that w.h.p. produces a solution of value $\lambda_{\text{OPT}}/\operatorname{poly}\log n$ and constant congestion. We also show that for any values η, α, such that $\eta \cdot \alpha \leq O(\log\log n/\log\log\log n)$, no efficient algorithm can find an α-approximate solution with congestion η to basic-ICF unless NP $\subseteq$ ZPTIME($n^{\text{poly}\log n}$). We then turn to the more challenging group-ICF problem. It is easy to see that when no congestion is allowed, the ratio of the optimal fractional to the optimal integral solution can be as large as $\Omega(\sqrt{n})$ for group-ICF, even when $k = 2$. Moreover, even if we allow congestion $c - 1$, this ratio can still be as large as $\Omega(n^{1/c})$, as shown in the full version of the paper. We show that for any $0 < \eta \leq O(\log\log n)$ and any $\alpha = O\left(n^{1/2^{2\eta+3}}\right)$,

[1]To reduce EDP to group-ICF, make D disjoint copies of the EDP instance. For each $1 \leq i \leq k$, let S_i contain all copies of s_i and T_i contain all copies of t_i. If we can find λD paths for every group (S_i, T_i), then some copy of the EDP instance will contain a solution of value at least λk.

no efficient algorithm can find α-approximate solutions with congestion η for group-ICF unless NP $\subseteq$ DTIME($n^{O(\log\log n)}$). Given an optimal integral solution $\mathcal{P}$ to the group-ICF problem instance, let $D = \min_i \{\lfloor \lambda^* \cdot D_i \rfloor\}$ be the minimum number of paths connecting any pair (S_i, T_i) in this solution. Our hardness result only holds for the regime where $D << k$. We show that if $D > k \operatorname{poly}\log n$, then there is an efficient algorithm that finds a $(\operatorname{poly}\log n)$-approximate solution with constant congestion. The value of the solution is in fact $\lambda_{\mathrm{OPT}}/\operatorname{poly}\log n$, where λ_{OPT} is the value of the optimal fractional solution. Therefore, when we allow a constant congestion, the ratio of the optimal fractional to the optimal integral solution becomes only polylogarithmic if $D > k \operatorname{poly}\log n$.

Our Results and Techniques.

Our first result is an approximation algorithm for the basic-ICF problem.

Theorem 1 *There is an efficient randomized algorithm, that, given any instance of* basic-ICF*, w.h.p. produces an integral solution of value $\lambda_{OPT}/\operatorname{poly}\log n$ and constant congestion, where λ_{OPT} is the value of the optimal fractional solution with no congestion to the* ICF *instance.*

The main technical tool that our algorithm uses is a graph decomposition similar to the one proposed by Andrews [1]. Assume first that the value of the minimum cut in graph G is polylogarithmic. We can then define $\operatorname{poly}\log n$ new graphs $G_1, \ldots, G_r$, where for each $1 \leq j \leq r$, $V(G_j) = V(G)$, and the edges in graphs G_j form a partition of the edges in G. If the value of the minimum cut in G is large enough, we can furthermore ensure that the value of the minimum cut in each resulting graph G_j is $\Omega(\log^5 n)$. We can then use the algorithm of Rao and Zhou [24] to find $\lambda^* \cdot \sum_i D_i/\operatorname{poly}\log n$ paths connecting the source-sink pairs in each graph G_j separately. By appropriately choosing the subsets of source-sink pairs for each graph G_j to connect, we can obtain a polylogarithmic approximation for the basic-ICF problem instance.

Unfortunately, it is possible that the global minimum cut in graph G is small. Andrews [1] in his paper on the EDP problem, suggested to get around this difficulty as follows. Let $L = \operatorname{poly}\log n$ be a parameter. For any subset C of vertices in G, let $\operatorname{out}(C) = E(C, V \setminus C)$ be the set of edges connecting the vertices of C to the vertices of $V \setminus C$. We say that a subset C of vertices is a large cluster iff $|\operatorname{out}(C)| \geq L$, and otherwise we say that it is a small cluster. Informally, we say that cluster C has the bandwidth property iff we can send $1/|\operatorname{out}(C)|$ flow units between every pair of edges in $\operatorname{out}(C)$ with small congestion inside the cluster C. Finally, we say that C is a critical cluster iff it is a large cluster, and we are given a partition $\pi(C)$ of its vertices into small clusters, such that on the one hand, each cluster in $\pi(C)$ has the bandwidth property, and on the other hand, the graph obtained from $G[C]$ by contracting every cluster in $\pi(C)$ is an expander. The key observation is that if C is a critical cluster, then we can integrally route demands on the edges of $\operatorname{out}(C)$ inside C, by using standard algorithms for routing on expanders. The idea of Andrews is that we can use the critical clusters to "hide" the small cuts in graph G.

More specifically, the graph decomposition procedure of Andrews consists of two steps. In the first step, he constructs what we call a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of graph G. Here, $\mathcal{Q}$ is a collection of disjoint critical clusters and $\mathcal{J}$ is a collection of disjoint small clusters that have the bandwidth property, and $\mathcal{Q} \cup \mathcal{J}$ is a partition of $V(G)$. This partition is computed in a way that ensures that every cut separating any pair of clusters in $\mathcal{Q}$ is large, containing at least $\operatorname{poly}\log n$ edges, and moreover we can connect all edges in $\bigcup_{C \in \mathcal{J}} \operatorname{out}(C)$ to the edges of $\bigcup_{C \in \mathcal{Q}} \operatorname{out}(C)$ by paths that together only cause a small congestion.

Given a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$, Andrews then constructs a new graph H, whose vertices are $\{v_Q \mid Q \in \mathcal{Q}\}$, and every edge $e = (v_Q, v_{Q'})$ in H is mapped to a path P_e in G connecting some vertex of Q to some vertex of Q', such that the total congestion caused by the set $\{P_e \mid e \in E\}$ of paths in graph G is small. Moreover, graph H preserves, to within a polylogarithmic factor, all cuts separating the clusters of $\mathcal{Q}$ in graph G. In particular, the size of the global minimum cut in H is large, and any integral routing in graph H can be transformed into an integral routing in G. This reduces the original problem to the problem of routing in the new graph H. Since the size of the minimum cut in graph H is large, we can now apply the algorithm proposed above to graph H.

We revisit the Q-J decomposition and the construction of the graph H from [1], and obtain an improved construction with better parameters. In particular, it allows us to reduce the routing congestion to constant, and to reduce the powers of the logarithms in the construction parameters. The Q-J decomposition procedure of [1] uses the tree decomposition of Räcke [22] as a black box. We instead perform the decomposition directly, thus improving some of its parameters. We also design a new well-linked decomposition procedure that may be of independent interest.

Our next result shows that basic-ICF is hard to approximate, using a simple reduction from the Congestion Minimization problem.

Theorem 2 *Given an n-vertex graph G with unit edge capacities, a collection $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs, and integers c, D, such that $Dc \leq O(\frac{\log\log n}{\log\log\log n})$, no efficient algorithm can distinguish between the following two cases unless* NP $\subseteq$ ZPTIME($n^{\operatorname{poly}\log n}$)*: (i) There is a collection $\mathcal{P}$ of paths that causes congestion 1, with D paths connecting s_i to t_i for each $1 \leq i \leq k$; and (ii) Any collection $\mathcal{P}$ of paths, containing, for each $1 \leq i \leq k$, at least one path connecting s_i to t_i, causes congestion at least c.*

We then turn to the group-ICF problem, and prove that it is hard to approximate in the following theorem.

Theorem 3 *Suppose we are given an n-vertex graph $G = (V, E)$ with unit edge capacities, and a collection of pairs of vertex subsets $(S_1, T_1), \ldots, (S_k, T_k)$. Let c be any integer, $0 < c \leq O(\log\log n)$ and let $D = O\left(n^{1/2^{2c+3}}\right)$. Then unless* NP $\subseteq$ DTIME($n^{O(\log\log n)}$)*, no efficient algorithm can distinguish between the following two cases: (i) There is a collection $\mathcal{P}^*$ of paths that causes congestion 1, and contains, for every $1 \leq i \leq k$, D paths connecting the vertices of S_i to the vertices of T_i; and (ii) Any set $\mathcal{P}^*$ of paths, containing, for each $1 \leq i \leq k$, at least one path connecting a vertex of S_i to a vertex of T_i, causes congestion at least c.*

The proof of Theorem 3 establishes a connection between

group-ICF and the Machine Minimization Scheduling problem, and then follows the hardness of approximation proof of [13] for the scheduling problem. Finally, we show an approximation algorithm for the group-ICF problem.

Theorem 4 *Suppose we are given an instance of* group-ICF*, and let $D = \min_i \{\lambda_{OPT} \cdot D_i\}$ be the minimum amount of flow sent between any pair (S_i, T_i) in the optimal fractional solution. Assume further that $D \geq \Delta'$, where $\Delta' = k \operatorname{poly}\log n$ is a parameter whose value we set later. Then there is an efficient randomized algorithm that finds a solution of value $\lambda_{OPT}/\operatorname{poly}\log n$ with constant congestion for the* group-ICF *instance.*

We now give a high-level overview of the proof of Theorem 4. We say that a Q-J decomposition is good iff no flow path in the optimal fractional solution is contained in any small cluster in $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$. We show an algorithm that finds a $(\operatorname{poly}\log n)$-approximate solution with constant congestion for instances where a good Q-J decomposition is given. This algorithm is similar to the algorithm from Theorem 1 for basic-ICF. Therefore, if we succeed in finding a good Q-J decomposition for instance $(G, \mathcal{D})$, we would have been done. However, we do not know how to obtain a good Q-J decomposition directly, so our algorithm instead partitions the input graph into a number of sub-instances. Each sub-instance either admits a good Q-J decomposition or corresponds to what we call a split instance, which can be solved using the algorithm of [12] as a subroutine, together with standard randomized rounding procedure.

Organization.

We start with Preliminaries in Section 2, and present the improved *Q-J* decomposition together with the construction of the graph H in Section 3. We show an algorithm for basic-ICF in Section 4, and hardness of basic-ICF and group-ICF in Sections 5 and 6 respectively. An algorithm for group-ICF appears in Section 7.

2. PRELIMINARIES

In all our algorithmic results, we first solve the problem for the special case where all edge capacities are unit, and then extend our algorithms to general edge capacities. Therefore, in this section, we only discuss graphs with unit edge capacities.

2.1 Demands and Routing

Given any subset $S \subseteq V$ of vertices in graph G, denote by $\operatorname{out}_G(S) = E_G(S, V \setminus S)$. We omit the subscript G when clear from context. Let $\mathcal{P}$ be any collection of paths in graph G. We say that paths in $\mathcal{P}$ cause congestion η, iff for each edge $e \in E(G)$, the number of paths in $\mathcal{P}$ containing e is at most η.

Given a graph $G = (V, E)$, and a set $\mathcal{T} \subseteq V$ of terminals, a set $\mathcal{D}$ of demands is a function $\mathcal{D} : \mathcal{T} \times \mathcal{T} \to \mathbb{R}^+$, that specifies, for each unordered pair $t, t' \in \mathcal{T}$ of terminals, a demand $D(t, t')$. We say that a set $\mathcal{D}$ of demands is γ-restricted, iff for each terminal $t \in \mathcal{T}$, the total demand $\sum_{t' \in \mathcal{T}} D(t, t') \leq \gamma$. Given any partition $\mathcal{G}$ of the terminals in $\mathcal{T}$, we say that a set $\mathcal{D}$ of demands is $(\gamma, \mathcal{G})$-restricted iff for each group $U \in \mathcal{G}$, $\sum_{t \in U} \sum_{t' \in \mathcal{T}} D(t, t') \leq \gamma$. We say that a demand set $\mathcal{D}$ is *integral* iff $D(t, t')$ is integral for all $t, t' \in \mathcal{T}$.

Given any set $\mathcal{D}$ of demands, a *fractional routing* of $\mathcal{D}$ is a flow F, where each pair $t, t' \in \mathcal{T}$, of terminals sends $D(t, t')$ flow units to each other. Given an integral set $\mathcal{D}$ of demands, an *integral* routing of $\mathcal{D}$ is a collection $\mathcal{P}$ of paths, that contains $D(t, t')$ paths connecting each pair (t, t') of terminals. The congestion of this integral routing is the congestion caused by the set $\mathcal{P}$ of paths in G. Any matching M on the set $\mathcal{T}$ of terminals defines an integral 1-restricted set $\mathcal{D}$ of demands, where $D(t, t') = 1$ if $(t, t') \in M$, and $D(t, t') = 0$ otherwise. We do not distinguish between the matching M and the corresponding set $\mathcal{D}$ of demands.

Given any two subsets V_1, V_2 of vertices, we denote by $F : V_1 \rightsquigarrow_\eta V_2$ a flow from the vertices of V_1 to the vertices of V_2 where each vertex in V_1 sends one flow unit, and the congestion due to F is at most η. Similarly, we denote by $\mathcal{P} : V_1 \rightsquigarrow_\eta V_2$ a collection of paths $\mathcal{P} = \{P_v \mid v \in V_1\}$, where each path P_v originates at v and terminates at some vertex of V_2, and the paths in $\mathcal{P}$ cause congestion at most η. We define flows and path sets between subsets of edges similarly. For example, given two collections E_1, E_2 of edges of G, we denote by $F : E_1 \rightsquigarrow_\eta E_2$ a flow that causes congestion at most η in G, where each flow-path has an edge in E_1 as its first edge, and an edge in E_2 as its last edge, and moreover each edge in E_1 sends one flow unit (notice that it is then guaranteed that each edge in E_2 receives at most η flow units due to the bound on congestion). We will often be interested in a scenario where we are given a subset $S \subseteq V(G)$ of vertices, and $E_1, E_2 \subseteq \operatorname{out}(S)$. In this case, we say that a flow $F : E_1 \rightsquigarrow_\eta E_2$ is *contained* in S, iff for each flow-path P in F, all edges of P belong to $G[S]$, except for the first and the last edges that belong to $\operatorname{out}(S)$. Similarly, we say that a set $\mathcal{P} : E_1 \rightsquigarrow_\eta E_2$ of paths is contained in S, iff all inner edges on paths in $\mathcal{P}$ belong to $G[S]$.

Edge-Disjoint Paths.

We use the algorithm of [12] for EDP, summarized in the following theorem.

Theorem 5 ([12]) *Let G be any graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. Assume further that there is a multicommodity flow where the pairs in $\mathcal{M}$ altogether send* OPT *flow units to each other, with no congestion, and at most one flow unit is sent between each pair. Then there is an efficient randomized algorithm that w.h.p. finds a collection $\mathcal{P}$ of paths, connecting at least $\mathsf{OPT}/\alpha_{\text{EDP}}$ of the demand pairs, such that the congestion of $\mathcal{P}$ is at most $\eta_{\text{EDP}} = 14$, where $\alpha_{\text{EDP}} = \operatorname{poly}\log k$.*

2.2 Sparsest Cut and the Flow-Cut Gap

Given a graph $G = (V, E)$ with a subset $\mathcal{T}$ of vertices called terminals, the *sparsity* of any partition (A, B) of V is $\frac{|E(A,B)|}{\min\{|A \cap \mathcal{T}|, |B \cap \mathcal{T}|\}}$. The goal of the sparsest cut problem is to find a partition (A, B) of V with minimum sparsity. Arora, Rao and Vazirani [7] have shown an $O(\sqrt{\log k})$-approximation algorithm for the sparsest cut problem. We denote by $\mathcal{A}_{\text{ARV}}$ this algorithm and by $\alpha_{\text{ARV}}(k) = O(\sqrt{\log k})$ its approximation factor.

Sparsest cut is the dual of the Maximum Concurrent Flow problem, where for each pair (t, t') of terminals, the demand $D(t, t') = 1/k$. The maximum possible ratio, in any graph, between the value of the minimum sparsest cut and the value

λ of the maximum concurrent flow, is called the *flow-cut gap*. The flow-cut gap in undirected graphs, that we denote by $\beta_{\mathrm{FCG}}(k)$ throughout the paper, is $\Theta(\log k)$ [20, 16, 21, 8]. In particular, if the value of the sparsest cut in graph G is α, then every pair of terminals can send at least $\frac{\alpha}{k\beta_{\mathrm{FCG}}(k)}$ flow units to each other simultaneously with no congestion. Moreover, any 1-restricted set $\mathcal{D}$ of demands on the set $\mathcal{T}$ of terminals can be fractionally routed with congestion at most $2\beta_{\mathrm{FCG}}(k)/\alpha$ in G.

2.3 Well-linkedness

Given any subset $S \subseteq V$ of vertices, we say that S is α-*well-linked*, iff for any partition (A, B) of S, if we denote $\mathcal{T}_A = \mathrm{out}(S) \cap \mathrm{out}(A)$ and $\mathcal{T}_B = \mathrm{out}(S) \cap \mathrm{out}(B)$, then $|E(A, B)| \geq \alpha \cdot \min\{|\mathcal{T}_A|, |\mathcal{T}_B|\}$.

Given a subset S of vertices, we can sub-divide every edge $e \in \mathrm{out}(S)$ by a vertex z_e, and set $\mathcal{T}' = \{z_e \mid e \in \mathrm{out}(S)\}$. Let G_S be the sub-graph of the resulting graph induced by $S \cup \mathcal{T}'$. Then S is α-well-linked (for $\alpha \leq 1$) in G iff the value of the sparsest cut in graph G_S for the set $\mathcal{T}'$ of terminals is at least α. In particular, if S is α-well-linked, then any 1-restricted set $\mathcal{D}$ of demands on $\mathrm{out}(S)$ can be fractionally routed inside S with congestion at most $2\beta_{\mathrm{FCG}}(k)/\alpha$.

Similarly, given any graph $G = (V, E)$ with a subset $\mathcal{T}$ of vertices called terminals, we say that G is α-well-linked for $\mathcal{T}$ iff for any partition (A, B) of V, $|E_G(A, B)| \geq \alpha \cdot \min\{|\mathcal{T} \cap A|, |\mathcal{T} \cap B|\}$.

Let $L = \mathrm{poly}\log n$ be a parameter to be fixed later. (We use different values of L in different algorithms.) We say that a cluster $X \subseteq V(G)$ is *small* iff $|\mathrm{out}(X)| \leq L$, and we say that it is *large* otherwise.

A useful tool in graph routing algorithms is a well-linked decomposition [10, 22]. This is a procedure that, given any subset S of vertices, produces a partition $\mathcal{W}$ of S into well-linked subsets. In the following theorem we describe a new well-linked decomposition. In addition to the standard properties guaranteed by well-linked decompositions, we obtain a collection of paths connecting the edges of $\bigcup_{R \in \mathcal{W}} \mathrm{out}(R)$ to the edges of $\mathrm{out}(S)$, with a small congestion. The proof of the following theorem appears in the full version of the paper.

Theorem 6 (Extended well-linked decomposition) *There is an efficient algorithm, that, given any set S of vertices, with $|\mathrm{out}(S)| = k'$, produces a partition $\mathcal{W}$ of S with the following properties.*

- *For each set $R \in \mathcal{W}$, $|\mathrm{out}(R)| \leq k'$. If R is a large cluster, then it is $\alpha_W = \Omega\left(1/\log^{1.5} n\right)$-well-linked. If R is a small cluster, then it is $\alpha_S = \Omega\left(1/(\log\log n)^{1.5}\right)$-well-linked.*
- *Let $E^* = \left(\bigcup_{R \in \mathcal{W}} \mathrm{out}(R)\right) \setminus \mathrm{out}(S)$. Then we can efficiently find a set $N = \{\tau_e \mid e \in E^*\}$ of paths called* tendrils *contained in $G[S]$, where tendril τ_e connects edge e to some edge of $\mathrm{out}(S)$, each edge in $\mathrm{out}(S)$ participates in at most one tendril, and the total congestion caused by N is at most 3.*
- *$|E^*| \leq 0.4|\mathrm{out}(S)|$.*

The Grouping Technique.

The grouping technique was first introduced by Chekuri, Khanna and Shepherd [9], and it is widely used in algorithms for network routing [10, 24, 1], in order to boost network connectivity and well-linkedness parameters. We summarize it in the following theorem.

Theorem 7 *Suppose we are given a connected graph $G = (V, E)$, with weights $w(v)$ on vertices $v \in V$, and a parameter p. Assume further that for each $v \in V$, $0 \leq w(v) \leq p$, and $\sum_{v \in V} w(v) \geq p$. Then we can find a partition $\mathcal{G}$ of the vertices in V, and for each group $U \in \mathcal{G}$, find a tree $T_U \subseteq G$ containing all vertices of U, such that for each group $U \in \mathcal{G}$, $p \leq w(U) \leq 3p$, where $w(U) = \sum_{v \in U} w(v)$, and the trees $\{T_U\}_{U \in \mathcal{G}}$ are edge-disjoint.*

2.4 Bandwidth Property and Critical Clusters

Given a graph G, and a subset S of vertices of G we say that the *modified bandwidth condition* holds for S, iff S is α_{BW}-well-linked if it is a large cluster, and it is α_S-well-linked if it is a small cluster, where $\alpha_S = \Omega\left(1/(\log\log n)^{1.5}\right)$, and $\alpha_{\mathrm{BW}} = \alpha_W \cdot \alpha_S = \Omega\left(\frac{1}{(\log n \log\log n)^{1.5}}\right)$. For simplicity, we will use "bandwidth property" instead of "modified bandwidth property" from now on.

Given a subset S of vertices of G, and a partition π of S, let H_S be the following graph: start with $G[S]$, and contract each cluster $C \in \pi$ into a super-node v_C. Set the weight $w(v_C)$ of v_C to be $|\mathrm{out}_G(C)|$ (notice that the weight takes into account all edges incident on C, including those in $\mathrm{out}(S)$). We use the parameter $\lambda = \frac{\alpha_{\mathrm{BW}}}{8\alpha_{\mathrm{ARV}}(n)} = \Omega\left(\frac{1}{\log^2 n \cdot (\log\log n)^{1.5}}\right)$.

Definition 1 *Given a subset S of vertices of G and a partition π of S, we say that (S, π) has the weight property with parameter λ', iff for any partition (A, B) of $V(H_S)$, $|E_{H_S}(A, B)| \geq \lambda' \cdot \min\left\{\sum_{v \in A} w(v), \sum_{v \in B} w(v)\right\}$. If the weight property holds for the parameter $\lambda = \lambda'$, then we simply say that (S, π) has the weight property.*

Definition 2 *Given a subset S of vertices and a partition π of S, we say that S is a* critical cluster *iff (1) S is a large cluster and it has the bandwidth property; (2) Every cluster $R \in \pi$ is a small cluster and it has the bandwidth property; and (3) (S, π) has the weight property. Additionally, if $S = \{v\}$, and the degree of v is greater than L, then we also say that S is a critical cluster.*

Let $\eta^* = \frac{2\beta_{\mathrm{FCG}}(L)}{\alpha_S} = O((\log\log n)^{2.5})$. We say that a cut $(S, \overline{S})$ in G is *large* iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$. Note that we somewhat abuse the notation: We will say that a cluster S is large iff $|\mathrm{out}(S)| > L$, but we say that a cut $(S, \overline{S})$ is large iff $|E(S, \overline{S})| \geq \frac{L}{4\eta^*}$.

In the next lemma we show that if we are given any large cluster S that has the bandwidth property, then we can find a critical sub-cluster Q of S. Moreover, there is a subset of at least $L/4$ edges of $\mathrm{out}(Q)$ that can be routed to the edges of $\mathrm{out}(S)$. One can prove a similar lemma using the Räcke decomposition as a black-box. Since we use slightly different parameters in the definitions of small and critical clusters, we prove the lemma directly.

Lemma 1 *Let S be any large cluster that has the bandwidth property. Then we can efficiently find a critical cluster $Q \subseteq S$, a subset $E_Q \subseteq \text{out}(Q)$ of $L/4$ edges, and a set $\mathcal{P}_Q : E_Q \rightsquigarrow_{\eta^*} \text{out}(S)$ of paths, which are contained in $S \setminus Q$, such that for each edge $e \in \text{out}(S)$, at most one path of $\mathcal{P}_Q$ terminates at e.*

PROOF. Let G' be a graph obtained from G as follows: subdivide every edge $e \in \text{out}(S)$ with a vertex v_e, and let $T' = \{v_e \mid e \in \text{out}(S)\}$. Graph G' is the sub-graph of G induced by $T' \cup S$.

Throughout the algorithm, we maintain a collection π of disjoint subsets of vertices of S, together with a corresponding contracted graph Z, which is obtained from graph G', by contracting every cluster $C \in \pi$ into a super-node v_C. We say that π is a *good collection of clusters*, iff each cluster $C \in \pi$ is small and has the bandwidth property. The value $W(\pi)$ of the collection π of clusters is the number of edges in the corresponding contracted graph Z. We notice that some vertices of S may not belong to any cluster in π. Our initial collection is $\pi = \emptyset$.

We say that a cluster $S' \subseteq S$ is *canonical* for the collection π iff for every cluster $C \in \pi$, either $C \subseteq S'$, or $C \cap S' = \emptyset$.

Throughout the algorithm, we also maintain an active large cluster $S' \subseteq S$ (the initial cluster $S' = S$). We will ensure that S' is canonical w.r.t. the current collection π of good clusters, and it has the bandwidth property. We perform a number of iterations. In each iteration, one of the following three things happens: we either find a new good collection π' of clusters, with $W(\pi') < W(\pi)$, or find a critical cluster Q as required, or select a sub-cluster $S'' \subsetneq S'$ as our next active cluster. In the latter case, we will guarantee that S'' is canonical for the current collection π of good clusters, and it has the bandwidth property. An execution of an iteration is summarized in the next lemma, whose proof appears in the full version. The proof uses arguments similar in spirit to the analysis of the Räcke decomposition [22].

Lemma 2 *Let π be a good collection of clusters, and let $S' \subseteq S$ be a large cluster with the bandwidth property, such that S' is canonical for π. Assume additionally that there is a set $E_{S'} \subseteq \text{out}(S')$ of $L/4$ edges, and a set $\mathcal{P}_{S'} : E_{S'} \rightsquigarrow_{\eta^*} \text{out}(S)$ of paths in graph G, contained in $S \setminus S'$, where each edge in $\text{out}(S)$ is an endpoint of at most one path. Then there is an efficient algorithm, whose output is one of the following:*

- *Either a good collection π' of clusters with $W(\pi') < W(\pi)$.*
- *Or establishes that S' is a critical cluster, by computing, if $|S'| > 1$, a partition π^* of S' into small clusters that have bandwidth property, such that (S', π^*) has the weight property.*
- *Or a sub-cluster $S'' \subsetneq S'$, such that S'' is large, canonical for π, has the bandwidth property, and there is a set $E_{S''} \subseteq \text{out}(S'')$ of $L/4$ edges, and a set $\mathcal{P}_{S''} : E_{S''} \rightsquigarrow_{\eta^*} \text{out}(S)$ of paths in graph G, contained in $S \setminus S''$, where each edge in $\text{out}(S)$ is an endpoint of at most one path.*

We now complete the proof of Lemma 1. We start with $S' = S$ and an initial collection $\pi = \emptyset$. We then iteratively apply Lemma 2 to the current cluster S' and the current partition π. If the lemma returns a good collection π' of clusters, whose value $W(\pi')$ is smaller than the value $W(\pi)$ of π, then we replace π with π', set the current active cluster to be $S' = S$, and continue. Otherwise, if it returns a sub-cluster $S'' \subsetneq S'$, then we replace S' with S'' as the current active cluster and continue. Finally, if it establishes that S' is a critical cluster, then we return S', π^*, the set $E_{S'}$ of edges, and the collection $\mathcal{P}_{S'}$ of paths. It is easy to verify that the algorithm terminates in polynomial time: we partition the algorithm execution into phases. A phase starts with some collection π of clusters, and ends when we obtain a new collection π' with $W(\pi') < W(\pi)$. Clearly, the number of phases is bounded by $|E(G)|$. In each phase, we perform a number of iterations, where in each iteration we start with some active cluster $S' \subseteq S$, and replace it with another cluster $S'' \subsetneq S'$. Therefore, the number of iterations in each phase is bounded by n. □

Suppose we are given a collection $\mathcal{C}$ of disjoint vertex subsets in graph G. We say that a cut (A, B) in graph G is *canonical* w.r.t. $\mathcal{C}$, iff for each $C \in \mathcal{C}$, either $C \subseteq A$, or $C \subseteq B$. We say that it is a non-trivial canonical cut, iff both A and B contain at least one cluster in $\mathcal{C}$.

2.5 Routing across Small and Critical Clusters

Following the ideas of Andrews [1], we will treat critical clusters as contracted nodes and solve a routing problem in the resulting contracted graph. To be able to do so, we need to ensure that, for any small or critical cluster S, demands between edges in $\text{out}(S)$ can be routed with low congestion inside S.

We use the following theorem from [12] to route demands across small clusters.

Theorem 8 *Let G be any graph and $\mathcal{T}$ any subset of k vertices called terminals, such that G is α-well-linked for $\mathcal{T}$. Then we can efficiently find a partition $\mathcal{G}$ of the terminals in $\mathcal{T}$ into groups of size $\frac{\text{poly}\log k}{\alpha}$, such that, for any $(1, \mathcal{G})$-restricted set $\mathcal{D}$ of demands on $\mathcal{T}$, there is an efficient randomized algorithm that w.h.p. finds an integral routing of $\mathcal{D}$ in G with edge congestion at most 15.*

Suppose we are given a small cluster S that has the bandwidth property. Since $|\text{out}(S)| \leq \text{poly}\log n$, and S is α_S-well-linked, we can use Theorem 8 to find a partition $\mathcal{G}_S$ of the edges of $\text{out}(S)$ into sets of size $\text{poly}\log\log n$, such that any $(1, \mathcal{G}_S)$-restricted set $\mathcal{D}$ of demands can be integrally routed inside S with congestion 15 w.h.p.

Observation 1 *Let $\mathcal{G}$ be any partition of the set $\mathcal{T}$ of terminals, and let $\mathcal{D}$ be any set of $(\gamma, \mathcal{G})$-restricted integral demands. Then we can efficiently find 4γ sets $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of $(1, \mathcal{G})$-restricted integral demands, such that any routing of the demands in set $\bigcup_{i=1}^{4\gamma} \mathcal{D}_i$ gives a routing of the demands in $\mathcal{D}$ with the same congestion, and moreover, if the former routing is integral, so is the latter.*

PROOF. Let $\mathcal{G} = \{\mathcal{T}_1, \ldots, \mathcal{T}_r\}$. Our first step is to modify the set $\mathcal{D}$ of demands, so that it does not contain demand pairs that belong to the same set $\mathcal{T}_i$. Specifically, for every pair $(u, v) \in \mathcal{D}$, where $u, v \in \mathcal{T}_i$ for some $1 \leq i \leq r$, we replace the demand (u, v) with a pair of demands (u, x), (v, x), where x is any vertex in set $\mathcal{T}_{i+1}$ (if $i = r$, then x is any vertex in $\mathcal{T}_1$). Let $\mathcal{D}'$ be the resulting set of demands.

Clearly, any routing of $\mathcal{D}'$ gives a routing of $\mathcal{D}$ with the same congestion, and if the routing of $\mathcal{D}'$ is integral, so is the corresponding routing of $\mathcal{D}$. It is also easy to see that $\mathcal{D}'$ is $(2\gamma, \mathcal{G})$-restricted.

Our second step is to decompose $\mathcal{D}'$ into 4γ demand sets $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$, such that each set $\mathcal{D}_j$ of demands is $(1, \mathcal{G})$-restricted, and $\bigcup_{j=1}^{4\gamma} \mathcal{D}_j = \mathcal{D}'$. We construct a multi-graph graph H with vertices $v_1, \ldots, v_r$ corresponding to the groups $\mathcal{T}_1, \ldots, \mathcal{T}_r$ of $\mathcal{G}$. For every pair $(u, v) \in \mathcal{D}'$, with $u \in \mathcal{T}_i$, $v \in \mathcal{T}_j$, we add an edge (i, j) to graph H. Finding the decomposition $\mathcal{D}_1, \ldots, \mathcal{D}_{4\gamma}$ of the set $\mathcal{D}'$ of demands then amounts to partitioning the edges of H into 4γ matchings. Since the maximum vertex degree in H is at most 2γ, such a decomposition can be found by a simple greedy algorithm. □

Combining Theorem 8 with Observation 1, we get the following corollary for routing across small clusters.

Corollary 1 *Given any small cluster S that has the bandwidth property, we can efficiently find a partition $\mathcal{G}_S$ of the edges of $\mathrm{out}(S)$ into groups of size at most $z = \mathrm{poly}\log\log n$, such that, for any $\gamma \geq 1$, given any $(\gamma, \mathcal{G}_S)$-restricted set $\mathcal{D}$ of demands on the edges of $\mathrm{out}(S)$, there is an efficient randomized algorithm, that w.h.p. finds an integral routing of $\mathcal{D}$ inside S with congestion at most 60γ.*

The following theorem gives an efficient algorithm for integral routing across critical clusters. The proof of this theorem appears in the full version.

Theorem 9 *Suppose we are given any cluster S, together with a partition π of S into small clusters, such that every cluster $C \in \pi$ is $\alpha_S/3$-well-linked, and (S, π) has the weight property with parameter $\lambda/3$. Then we can efficiently find a partition $\mathcal{G}$ of the edges of $\mathrm{out}(S)$ into groups of size at least $Z = O(\log^4 n)$ and at most $3Z$, such that, for any set $\mathcal{D}$ of $(1, \mathcal{G})$-restricted demands on $\mathrm{out}(S)$, there is an efficient randomized algorithm that w.h.p. routes $\mathcal{D}$ integrally in $G[S]$ with congestion at most 721.*

3. GRAPH DECOMPOSITION AND SPLITTING

In this section we present the main technical tools that our algorithms use: the Q-J decomposition, the construction of the graph H, and the splitting of H into sub-graphs. We start with the Q-J decomposition.

3.1 Graph Decomposition

We assume that we are given a non-empty collection $\mathcal{Q}_0$ of disjoint critical clusters in graph G, with the following property: if (A, B) is any non-trivial canonical cut in graph G w.r.t. $\mathcal{Q}_0$, then it is a large cut. For motivation, consider the basic-ICF problem, and assume that every cut separating the terminals in $\mathcal{T}$ is a large cut. Then we can set $\mathcal{Q}_0 = \{\{t\} \mid t \in \mathcal{T}\}$. For the group-ICF problem, we will compute $\mathcal{Q}_0$ differently, by setting $\mathcal{Q}_0 = \{Q\}$ where Q is an arbitrary critical cluster in G.

Suppose we are given a collection $\mathcal{Q}$ of disjoint critical clusters, and a collection $\mathcal{J}$ of disjoint small clusters, such that $\mathcal{Q} \cup \mathcal{J}$ is a partition of $V(G)$. Let $E^Q = \bigcup_{Q \in \mathcal{Q}} \mathrm{out}(Q)$, and let $E^J = \left(\bigcup_{J \in \mathcal{J}} \mathrm{out}(J)\right) \setminus E^Q$. We say that $(\mathcal{Q}, \mathcal{J})$ is a valid Q-J decomposition, iff $\mathcal{Q}_0 \subseteq \mathcal{Q}$, and:

- P1. Every cluster $J \in \mathcal{J}$ is a small cluster with the bandwidth property, and every cluster $Q \in \mathcal{Q}$ is a critical cluster.
- P2. There is a set $N = \left\{\tau_e \mid e \in E^J\right\}$ of paths, called *tendrils*, where path τ_e connects e to some edge in E^Q, each edge in E^Q is an endpoint of at most one tendril, and the total congestion caused by N is at most 3. Moreover, the tendrils do not use edges $e = (u, v)$ where both u and v belong to clusters in $\mathcal{Q}$.
- P3. If $(S, \overline{S})$ is any cut in graph G, which is non-trivial and canonical w.r.t. $\mathcal{Q}$, then it is a large cut.

We refer to the clusters in $\mathcal{Q}$ as the Q-clusters, and to the clusters in $\mathcal{J}$ as the J-clusters. We note that Andrews [1] has (implicitly) defined the Q-J decomposition, and suggested an algorithm for constructing it, by using the graph decomposition of Räcke [22] as a black-box. The Räcke decomposition however gives very strong properties - stronger than one needs to construct a Q-J decomposition. We obtain a Q-J decomposition with slightly better parameters by performing the decomposition directly, instead of using the Räcke decomposition as a black-box. For example, the tendrils in N only cause a constant congestion in our decomposition, instead of a logarithmic one, the well-linkedness of the J-clusters is $\mathrm{poly}\log\log n$ instead of $\mathrm{poly}\log n$, and we obtain a better relationship between the parameter L and the size of the minimum cut separating the Q-clusters. The algorithm for finding a Q-J decomposition is summarized in the next theorem.

Theorem 10 *There is an efficient algorithm, that, given any graph G, and a set $\mathcal{Q}_0$ of disjoint critical clusters, such that any non-trivial canonical cut w.r.t. $\mathcal{Q}_0$ in G is large, produces a valid Q-J decomposition of G.*

Proof. For each edge $e \in E^J$, if v is the endpoint of the tendril $\tau(e)$ that belongs to some Q-cluster in $\mathcal{Q}$, then we say that v is the *head* of the tendril $\tau(e)$. We also set $Q^* = \bigcup_{Q \in \mathcal{Q}} Q$.

We build the clusters in $\mathcal{Q}$ gradually. The algorithm performs a number of iterations. We start with $\mathcal{Q} = \mathcal{Q}_0$, and in each iteration, we add a new critical cluster Q to $\mathcal{Q}$. In the last iteration, we produce the set $\mathcal{J}$ of the J-clusters, and their tendrils, as required.

We now proceed to describe an iteration. Let $\mathcal{Q}$ be the set of current Q-clusters, and let $Q^* = \bigcup_{Q \in \mathcal{Q}} Q$. Let $S_0 = V \setminus Q^*$.

We start by performing the extended well-linked decomposition of the set S_0 of vertices, using Theorem 6. Let $\mathcal{W}$ be the resulting decomposition, and N the corresponding set of tendrils. If all sets in $\mathcal{W}$ are small, then we set $\mathcal{J} = \mathcal{W}$, and finish the algorithm, so the current iteration becomes the last one. The output is $(\mathcal{Q}, \mathcal{J})$, and the final set of tendrils is N. We will later show that it has all required properties. Assume now that $\mathcal{W}$ contains at least one large cluster, and denote it by S. Let N_S be the set of tendrils originating at edges of $\mathrm{out}(S)$.

We use Lemma 1 to find a critical sub-cluster $Q \subseteq S$, together with the subset $E_Q \subseteq \mathrm{out}(Q)$ of $L/4$ edges, and the set $\mathcal{P}_S : E_Q \rightsquigarrow_{\eta^*} \mathrm{out}(S)$ of paths, that are contained in $S \setminus Q$. Let $\mathcal{P}'_S : E_Q \rightsquigarrow_{\eta^*} \mathrm{out}(S_0)$ be a collection of paths obtained by concatenating the paths in $\mathcal{P}_S$ with the set N_S of tendrils

originating at the edges of out(S). Notice that each edge of out(S_0) serves as an endpoint of at most one such path, and $|\mathcal{P}'_S| = L/4$. We then add Q to $\mathcal{Q}$, and continue to the next iteration. This concludes the description of an iteration. Consider the final collections $\mathcal{Q}, \mathcal{J}$ of clusters produced by the algorithm. It is immediate to see that Properties (P1)–(P2) hold for it. We only need to establish Property (P3).

Consider any cut $(S, \overline{S})$ in graph G, such that for each cluster $Q \in \mathcal{Q}$, either $Q \subseteq S$, or $Q \subseteq \overline{S}$, and assume that both S and $\overline{S}$ contain at least one cluster in $\mathcal{Q}$. We say that the vertices of S are red and the vertices of $\overline{S}$ are blue.

If both S and $\overline{S}$ contain clusters from $\mathcal{Q}_0$, then the cut $(S, \overline{S})$ must be large by our initial assumption. Assume w.l.o.g. that all clusters in $\mathcal{Q}_0$ are red. Let Q be the first cluster that has been added to $\mathcal{Q}$ over the course of the algorithm, whose vertices are blue. Recall that we have a set $\mathcal{P}'_Q$ of $L/4$ paths connecting the edges of out(Q) to the edges of out(S_0) with congestion at most η^*. Therefore, there must be at least $\frac{L}{4\eta^*}$ edges in the cut, so $(S, \overline{S})$ is a large cut. This concludes the proof of Theorem 10. □

Given a valid Q-J decomposition, it is not hard to construct a graph H with the desired properties. The following theorem mostly follows the construction of [1], with some minor changes. We defer its proof to the full version.

Theorem 11 *Suppose we are given a valid Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ for graph G. Then there is an efficient randomized algorithm to construct a graph H with $V(H) = \{v_C \mid C \in \mathcal{Q}\}$, and for each edge $e = (v_{C_1}, v_{C_2}) \in E(H)$, define a path P_e in graph G, connecting some vertex of C_1 to some vertex of C_2, such that for some value $\alpha^* = O(\log^4 n \operatorname{poly}\log\log n)$, the following properties hold w.h.p. for graph H:*

- *C1. For every cut (A, B) in graph H, there is a cut (A', B') in graph G, such that for each $Q \in \mathcal{Q}$, if $v_Q \in A$ then $Q \subseteq A'$, and if $v_Q \in B$ then $Q \subseteq B'$, and $|E_G(A', B')| \leq \alpha^* \cdot |E_H(A, B)|$.*
- *C2. The value of the minimum cut in H is at least $\frac{L}{\alpha^*}$.*
- *C3. The paths in set $\mathcal{P}_H^E = \{P_e \mid e \in E(H)\}$ cause a constant congestion in graph G.*
- *C4. For each critical cluster $C \in \mathcal{Q}$, let $\mathcal{G}_C$ be the grouping of the edges of* out(C) *given by Theorem 9. Then for each group $U \in \mathcal{G}_C$, at most two paths in $\mathcal{P}_H^E$ contain an edge of U as their first or last edge.*

3.2 Graph Splitting

Once we compute the graph H, we can split it into graphs $H_1, \ldots, H_x$, as follows. For each $1 \leq j \leq x$, the set of vertices $V(H_j) = V(H)$. The sets of edges $E(H_1), \ldots, E(H_x)$ are constructed as follows. Each edge $e \in E(H)$ independently chooses an index $j \in \{1, \ldots, x\}$ uniformly at random. Edge e is then added to graph H_j, where j is the index chosen by e. We use the following theorem (a re-statement of Theorem 2.1 from [18]).

Theorem 12 ([18]) *Let $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ be any n-vertex graph with minimum cut value C. Assume that we obtain a subgraph $\mathbf{G}' = (\mathbf{V}, \mathbf{E}')$, by adding every edge $e \in \mathbf{E}$ with probability p to $\mathbf{E}'$, and assume further that $C \cdot p > 48 \ln n$. Then with probability at least $1 - O(1/n^2)$, for every partition (A, B) of $\mathbf{V}$, $|E_{\mathbf{G}'}(A, B)| \geq \frac{p|E_{\mathbf{G}}(A,B)|}{2}$.*

Therefore, if we select L so that $\frac{L}{x\alpha^*} > 48 \ln n$, then we can perform the graph splitting as described above, and from Theorem 12, for each $1 \leq j \leq x$, for each partition (A, B) of $V(H)$, $|E_{H_j}(A, B)| \geq \frac{|E_H(A,B)|}{2x}$ w.h.p.

4. AN ALGORITHM FOR BASIC-ICF

The goal of this section is to prove Theorem 1. We start by proving the theorem for the special case where all demands are uniform, and all edge capacities are unit. That is, we are given a subset $\mathcal{M} \subseteq \mathcal{T} \times \mathcal{T}$ of the terminal pairs, and for each pair $(t, t') \in \mathcal{M}$, $D(t, t') = D$, while all other demands are 0. We extend this algorithm to handle arbitrary demands and edge capacities in Section A of the Appendix. We set the parameter $L = \Theta(\log^{20} n \operatorname{poly}\log\log n)$, and we define its exact value later.

4.1 The Algorithm

Assume first that in the input instance G, any cut separating the set $\mathcal{T}$ of terminals has value at least L. We show in the next theorem, that in this case we can find an integral solution of value $\lambda_{\text{OPT}}/\operatorname{poly}\log n$ with constant congestion to instance $(G, \mathcal{D})$. The main idea is to use Theorem 11 to construct a graph H, where the initial set $\mathcal{Q}_0$ of critical clusters is $\mathcal{Q}_0 = \{\{t\} \mid t \in \mathcal{T}\}$. We then split H into $\operatorname{poly}\log n$ sub-graphs using the procedure outlined in Section 3, and use the algorithm of Rao and Zhou [24] to route a poly-logarithmic fraction of the demand pairs in each resulting subgraph. The proof of the next theorem appears later in Section 4.2.

Theorem 13 *Let G be an instance of* basic-ICF *with unit edge capacities and a set $\mathcal{D}$ of uniform demands over the set $\mathcal{T}$ of terminals. Assume that every cut separating the terminals in $\mathcal{T}$ has size at least $L = \Theta(\log^{20} n \operatorname{poly}\log\log n)$. Then there is an efficient randomized algorithm that w.h.p. finds an integral solution of value λ_{OPT}/β with constant congestion, where $\beta = O(\log^{26} n \operatorname{poly}\log\log n)$ and λ_{OPT} is the value of the optimal fractional solution.*

In general, graph G may contain small cuts that separate its terminals. We get around this problem as follows. For each subset $S \subseteq V$ of vertices, let $\mathcal{T}_S = S \cap \mathcal{T}$ be the subset of terminals contained in S. We say that S is a *good subset* iff (1) Any cut in graph $G[S]$ separating the terminals in $\mathcal{T}_S$ has value at least L; and (2) $|\text{out}(S)| \leq L \log k$. We first show that we can efficiently compute a good set S of vertices in graph G. We then decompose the set $\mathcal{D}$ of demands into two subsets: $\mathcal{D}_S$ containing the demands for all pairs contained in $\mathcal{T}_S$, and $\mathcal{D}'$ containing the demands for all other pairs. Next, we apply Theorem 13 to instance $(G[S], \mathcal{D}_S)$, obtaining a collection $\mathcal{P}'$ of paths, and solve the problem recursively on instance $(G, \mathcal{D}')$, obtaining a collection $\mathcal{P}''$ of paths. Our final step is to carefully combine the two sets of paths to obtain the final solution $\mathcal{P}$. We start with the following lemma that allows us to find a good subset S of vertices efficiently.

Lemma 3 *Let $(G, \mathcal{D})$ be a* basic-ICF *instance with uniform demands and unit edge capacities, and a set $\mathcal{T}$ of terminals, where $|\mathcal{T}| \leq k$. Then there is an efficient algorithm that*

either finds a good set $S \subseteq V(G)$ of vertices, or establishes that every cut (A, B) separating the terminals of $\mathcal{T}$ in G has value $|E_G(A, B)| \geq L$.

PROOF. We start with $S = V(G)$, and then perform a number of iterations. Let $G' = G[S]$, and let $\mathcal{T}_S = \mathcal{T} \cap S$. Let (A, B) be the minimum cut separating the terminals in $\mathcal{T}_S$ in graph G', and assume w.l.o.g. that $|A \cap \mathcal{T}_S| \leq |B \cap \mathcal{T}_S|$. If $|E_{G'}(A, B)| < L$, then we set $S = A$, and continue to the next iteration. Otherwise, we output S as a good set. (If $S = V(G)$, then we declare that every cut separating the terminals in $\mathcal{T}$ has value at least L.)

Clearly, if S is the final set that the algorithm outputs, then every cut in graph $G[S]$ separating the set $\mathcal{T} \cap S$ of terminals has value at least L. It only remains to show that $|\operatorname{out}_G(S)| \leq L \log k$. Since $|\mathcal{T}| \leq k$, and the number of terminals contained in set S goes down by a factor of at least 2 in every iteration, there are at most $\log k$ iterations. In each iteration, at most L edges are deleted. Therefore, $|\operatorname{out}_G(S)| \leq L \log k$. □

We use the following theorem, whose proof appears below in Section 4.3, to combine the solutions to the two sub-instances. In this theorem, we assume that we are given a good vertex set S, $\mathcal{T}_S = \mathcal{T} \cap S$, and $\mathcal{M}_S \subseteq \mathcal{M}$ is the subset of the demand pairs contained in S. We assume w.l.o.g. that $\mathcal{M}_S = \{(s_1, t_1), \ldots, (s_{k'}, t_{k'})\}$.

Theorem 14 *Suppose we are given a good vertex set S, and $\mathcal{M}_S \subseteq \mathcal{M}$ as above. Assume further that for each $1 \leq i \leq k'$, we are given a set $\mathcal{P}_i$ of N paths connecting s_i to t_i, such that all paths in set $\mathcal{P}' = \bigcup_{i=1}^{k'} \mathcal{P}_i$ are contained in $G[S]$, and set $\mathcal{P}'$ causes congestion at most γ in $G[S]$. Let $\mathcal{P}''$ be any set of paths in graph G, where each path in $\mathcal{P}''$ connects some pair $(s, t) \in \mathcal{M} \setminus \mathcal{M}_S$, and the congestion caused by the paths in $\mathcal{P}''$ is at most γ. Then we can efficiently find, for each $1 \leq i \leq k'$, a subset $\mathcal{P}_i^* \subseteq \mathcal{P}_i$ of at least $N - 2L\gamma \log n$ paths, and for each $P \in \mathcal{P}''$, find a path $\tilde{P}$ connecting the same pair of vertices as P, such that the total congestion caused by the set $\left(\bigcup_{i=1}^{k'} \mathcal{P}_i^*\right) \cup \left\{\tilde{P} \mid P \in \mathcal{P}''\right\}$ of paths is at most γ in graph G.*

We denote this algorithm by $\textsc{Reroute}(\mathcal{P}', \mathcal{P}'')$, and its output is denoted by $\tilde{\mathcal{P}}' = \bigcup_{i=1}^{k'} \mathcal{P}_i^*$, and $\tilde{\mathcal{P}}'' = \left\{\tilde{P} \mid P \in \mathcal{P}''\right\}$. We now complete the description of our algorithm, that we call RecursiveRouting.

We assume that we are given a graph G, a set $\mathcal{T}$ of at most k terminals, a set $\mathcal{M}$ of demand pairs, and a real number $D > 0$, such that for each pair $(t, t') \in \mathcal{M}$, we can send $\lambda_{\text{OPT}} \cdot D$ flow units simultaneously in G with no congestion. If no cut of size less than L separates the terminals of $\mathcal{T}$, then we use Theorem 13 to find a set $\mathcal{P}$ of paths and return $\mathcal{P}$. Otherwise, we find a good vertex set $S \subseteq V(G)$ using Lemma 3. Let $\mathcal{T}_S = \mathcal{T} \cap S$, $\mathcal{M}_S \subseteq \mathcal{M}$ the subset of pairs contained in $\mathcal{T}_S$, $\mathcal{M}' = \mathcal{M} \setminus \mathcal{M}_S$. We then apply Theorem 13 to $(G[S], \mathcal{M}_S, D)$ to obtain a collection $\mathcal{P}'$ of paths, and invoke RecursiveRouting on $(G, \mathcal{M}', D)$ to obtain a set $\mathcal{P}''$ of paths. Finally, we apply procedure ReRoute to sets $\mathcal{P}', \mathcal{P}''$ of paths to obtain the collections $\tilde{\mathcal{P}}', \tilde{\mathcal{P}}''$ of paths, and return $\mathcal{P} = \tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$.

Let β be the parameter from Theorem 13, and let γ be the congestion it guarantees. We can assume w.l.o.g. that $\frac{\lambda_{\text{OPT}}}{\beta} D \geq 4L\gamma \log n$, since otherwise $\lfloor \frac{\lambda_{\text{OPT}}}{4\beta L\gamma \log n} D \rfloor = 0$, and $\mathcal{P} = \emptyset$ is a (poly log)-approximate solution to the problem. We now prove that procedure RecursiveRouting produces a solution of value $\frac{\lambda_{\text{OPT}}}{4\beta}$ and congestion at most γ.

Lemma 4 *Let $\mathcal{P}$ be the output of procedure* RecursiveRouting. *Then for every pair $(s, t) \in \mathcal{M}$, at least $\lfloor \frac{\lambda_{OPT}}{4\beta} D \rfloor$ paths connect s to t in $\mathcal{P}$, and the paths in $\mathcal{P}$ cause congestion at most γ in G.*

PROOF. The proof is by induction on the recursion depth. In the base case, when no cut of size less than L separates the terminals of $\mathcal{T}$ in G, the correctness follows directly from Theorem 13.

Otherwise, consider the set $\mathcal{P}'$ of paths. For each pair $(s, t) \in \mathcal{M}_S$, let $\mathcal{P}'(s, t) \subseteq \mathcal{P}'$ be the subset of paths connecting s to t in $\mathcal{P}'$. From Theorem 13, we are guaranteed that $|\mathcal{P}'(s, t)| \geq \lfloor \frac{\lambda_{\text{OPT}}}{\beta} D \rfloor$ for each $(s, t) \in \mathcal{M}_S$, and the paths in $\mathcal{P}'$ cause congestion at most γ in $G[S]$.

Consider now the set $\mathcal{P}''$ of paths. For each pair $(s, t) \in \mathcal{M}'$, let $\mathcal{P}''(s, t) \subseteq \mathcal{P}''$ be the subset of paths connecting s to t in $\mathcal{P}''$. From the induction hypothesis, $|\mathcal{P}''(s, t)| \geq \lfloor \frac{\lambda_{\text{OPT}}}{4\beta} D \rfloor$ for all $(s, t) \in \mathcal{M}'$, and the paths in $\mathcal{P}''$ cause congestion at most γ in G.

Consider now the final set $\mathcal{P} = \tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$ of paths returned by the algorithm. From Theorem 14, the paths in $\mathcal{P}$ cause congestion at most γ, as required. For each pair $(s, t) \in \mathcal{M}_S$, the set $\tilde{\mathcal{P}}'$ of paths contains at least $\lfloor \frac{\lambda_{\text{OPT}}}{\beta} D \rfloor - 2L\gamma \log n \geq \lfloor \frac{\lambda_{\text{OPT}}}{4\beta} D \rfloor$ paths. For each pair $(s, t) \in \mathcal{M}_S$, if $\tilde{\mathcal{P}}''(s, t)$ is the subset of paths of $\tilde{\mathcal{P}}''$ connecting s to t, then $|\tilde{\mathcal{P}}''(s, t)| = |\mathcal{P}''(s, t)| \geq \lfloor \frac{\lambda_{\text{OPT}}}{4\beta} D \rfloor$, since each path in $\mathcal{P}''$ is replaced by a path connecting the same pair of vertices in $\tilde{\mathcal{P}}''$. Therefore, each pair $(s, t) \in \mathcal{M}$ is connected by at least $\lfloor \frac{\lambda_{\text{OPT}}}{4\beta} D \rfloor$ paths in $\mathcal{P}$. □

This completes the proof of Theorem 1 for uniform demands and unit edge capacities, except for the proofs of Theorems 13 and 14 that appear below. In Section A of the Appendix we extend this algorithm to arbitrary edge capacities and demands using standard techniques.

4.2 Proof of Theorem 13

We assume that we are given a collection $\mathcal{M} \subseteq \mathcal{T} \times \mathcal{T}$ of terminal pairs and a value $D > 0$, such that for each pair $(t, t') \in \mathcal{M}$, $D(t, t') = 1$, and all other demands are 0.

We start by applying Theorem 11 to graph G, where we use the threshold L from the statement of Theorem 13 for the definition of small clusters, and the initial set of critical clusters is $\mathcal{Q}_0 = \{\{t\} \mid t \in \mathcal{T}\}$. It is easy to see that each cluster in $\mathcal{Q}_0$ is a critical cluster, and any cut separating the clusters in $\mathcal{Q}_0$ in graph G is large. Let H be the resulting graph guaranteed by Theorem 11.

Since every terminal in $\mathcal{T}$ is mapped to a separate vertex of H, we can view $\mathcal{D}$ as a set of demands for graph H. We now focus on finding a solution to the ICF problem instance in graph H, and later transform it into a solution in the original graph G. We use the following theorem, due to Rao and Zhou [24].

Theorem 15 ([24]) *Let G' be any n-vertex graph, and let $\mathcal{M}' = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ be any set of source-sink pairs in G'. Assume further that the value of the global minimum cut in graph G' is at least $L_{\text{RZ}} = \Omega(\log^5 n)$, and there is a*

fractional multi-commodity flow, where for each $1 \le i \le k$, source s_i sends $f_i \le 1$ flow units to sink t_i, $\sum_{i=1}^{k} f_i = F$, and the flow causes no congestion in G'. Then there is an efficient randomized algorithm that w.h.p. finds a collection $\mathcal{M}^* \subseteq \mathcal{M}'$ of at least F/α_{RZ} demand pairs, and for each pair $(s,t) \in \mathcal{M}^*$, a path $P(s,t)$ connecting s to t in G', such that the paths in the set $\mathcal{P}^* = \{P(s,t) \mid (s,t) \in \mathcal{M}^*\}$ are edge-disjoint, and $\alpha_{\text{RZ}} = O(\log^{10} n)$.

Let $x = 8\alpha_{\text{RZ}} \cdot \log n = O(\log^{11} n)$. We set $L = 2\alpha^* \cdot x \cdot L_{\text{RZ}} = O(\log^{20} n \operatorname{poly}\log\log n)$.

We split graph H into x graphs $H_1, \ldots, H_x$, as follows. For each $1 \le i \le x$, we set $V(H_i) = V(H)$. In order to define the edge sets of graphs H_i, each edge $e \in E$, chooses an index $1 \le i \le x$ independently uniformly at random, and is then added to $E(H_i)$. This completes the definition of the graphs H_i. Given any partition (A,B) of the vertices of $V(H)$, let $\text{cut}_G(A,B)$ denote the value of the minimum cut $|E_G(A',B')|$ in graph G, such that for each $v_Q \in A$, $Q \subseteq A'$, and for each $v_Q \in B$, $Q \subseteq B'$. Recall that Theorem 11 guarantees that the size of the minimum cut in H is at least L/α^*, and for each partition (A,B) of $V(H)$, $\text{cut}_G(A,B) \le \alpha^* \cdot |E_H(A,B)|$. From Theorem 12, w.h.p., for each $1 \le i \le x$, we have that:

- The value of the minimum cut in H_i is at least $\frac{L}{2\alpha^* x} = L_{\text{RZ}}$.
- For any cut (A,B) of $V(H_i)$, $|E_{H_i}(A,B)| \ge \frac{\text{cut}_G(A,B)}{2x\alpha^*}$.

From now on we assume that both properties hold for each graph H_i. We then obtain the following observation, whose proof appears in the full version of the paper.

Observation 2 *For each $1 \le i \le x$, there is a fractional solution to the instance $(H_i, \mathcal{D})$ of* basic-ICF *of value $\frac{\lambda_{OPT}}{2x\alpha^*\beta_{\text{FCG}}}$ and no congestion.*

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs $H_1, \ldots, H_x$ in turn. For each $1 \le i \le x$, in the ith iteration we define a subset $\mathcal{M}_i \subseteq \mathcal{M}$ of pairs of terminals (that are not satisfied yet), and we define the set $\mathcal{D}_i$ of demands to be $D_i(t,t') = D$ if $(t,t') \in \mathcal{M}_i$, and $D_i(t,t') = 0$ otherwise. For the first iteration, $\mathcal{M}_1 = \mathcal{M}$. We now describe an execution of iteration $i \ge 1$.

Suppose we are given a set $\mathcal{M}_i$ of terminal pairs and a corresponding set $\mathcal{D}_i$ of demands. We construct a new collection $\mathcal{M}_i'$ of source-sink pairs, where the demand for each pair is 1, as follows. For each pair $(t,t') \in \mathcal{M}_i$, we add $N = \lfloor \frac{\lambda_{\text{OPT}}}{2x\alpha^*\beta_{\text{FCG}}} \cdot D \rfloor$ copies of the pair (t,t') to $\mathcal{M}_i'$. We then apply Theorem 15 to the resulting graph and the set $\mathcal{M}_i'$ of demand pairs. From Observation 2, there is a flow of value at least $F_i = N \cdot |\mathcal{M}_i| = |\mathcal{M}_i'|$ in the resulting graph. Therefore, from Theorem 15, w.h.p. we obtain a collection $\mathcal{P}_i$ of paths connecting the demand pairs in $\mathcal{M}_i$ with no congestion, and $|\mathcal{P}_i| \ge \frac{F_i}{\alpha_{\text{RZ}}} \ge \frac{N \cdot |\mathcal{M}_i|}{\alpha_{\text{RZ}}}$. We say that a pair $(t,t') \in \mathcal{M}_i$ of terminals is satisfied in iteration i, iff the number of paths in $\mathcal{P}_i$ connecting t to t' is at least $\frac{N}{2\alpha_{\text{RZ}}}$. We then let $\mathcal{M}_{i+1} \subseteq \mathcal{M}_i$ be the subset of terminal pairs that are not satisfied in iteration i. This concludes the description of our algorithm for routing on graph H. The key in its analysis is the following simple claim.

Claim 1 *For each $1 \le i \le x$, $|\mathcal{M}_{i+1}| \le \left(1 - \frac{1}{2\alpha_{\text{RZ}}}\right)|\mathcal{M}_i|$.*

PROOF. Let $\mathcal{M}_i^* \subseteq \mathcal{M}_i$ be the subset of demand pairs that are satisfied in iteration i. It is enough to prove that $|\mathcal{M}_i^*| \ge \frac{1}{2\alpha_{\text{RZ}}}|\mathcal{M}_i|$. Assume otherwise. A pair $(t,t') \in \mathcal{M}_i^*$ contributes at most N paths to $\mathcal{P}_i$, while a pair $(t,t') \in \mathcal{M}_i \setminus \mathcal{M}_i^*$ contributes less than $\frac{N}{2\alpha_{\text{RZ}}}$ paths. Therefore, if $|\mathcal{M}_i^*| < \frac{1}{2\alpha_{\text{RZ}}}|\mathcal{M}_i|$, then:

$$|\mathcal{P}_i| < |\mathcal{M}_i^*| \cdot N + |\mathcal{M}_i \setminus \mathcal{M}_i^*| \cdot \frac{N}{2\alpha_{\text{RZ}}} < \frac{N}{\alpha_{\text{RZ}}}|\mathcal{M}_i|,$$

a contradiction. □

Therefore, after $x = 8\alpha_{\text{RZ}} \cdot \log n$ iterations, we will obtain $\mathcal{M}_{x+1} = \emptyset$, and all demand pairs are satisfied. Recall that a demand pair is satisfied iff there are at least $\frac{N}{2\alpha_{\text{RZ}}} = \Omega\left(\frac{\lambda_{\text{OPT}}}{\alpha_{\text{RZ}} x \alpha^* \beta_{\text{FCG}}} \cdot D\right) = \Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{26} n \operatorname{poly}\log\log n} D\right)$ paths connecting them. Therefore, we have shown an integral solution to the ICF instance $(H, \mathcal{D})$ of value $\Omega\left(\frac{\lambda_{\text{OPT}}}{\log^{26} n \operatorname{poly}\log\log n}\right)$ and no congestion.

We now show how to obtain an integral solution to the ICF instance $(G, \mathcal{D})$, of the same value and constant congestion. Let $\mathcal{P}^*$ be the set of paths in graph H that we have obtained. We transform each path $P \in \mathcal{P}^*$ into a path P' connecting the same pair of terminals in graph G. Recall that all terminals in $\mathcal{T}$ are vertices in both G and H. For each edge $e = (v_Q, v_{Q'})$ on path P, we replace e with the path P_e, connecting some vertex $u \in Q$ to some vertex $u' \in Q'$, guaranteed by Property (C3) of graph H. Once we process all edges on all paths $P \in \mathcal{P}^*$, we obtain, for each cluster $Q \in \mathcal{Q}$, a set $\mathcal{D}_Q$ of demands on the edges of $\text{out}(Q)$, that need to be routed inside the cluster Q. From Property (C4), this set of demands must be $(2, \mathcal{G}_Q)$-restricted. Combining Observation 1 with Theorem 9, we obtain an efficient randomized algorithm that w.h.p. routes the set $\mathcal{D}_Q$ of demands integrally inside Q with constant congestion. For each path $P \in \mathcal{P}^*$, we can now combine the paths P_e for $e \in P$ with the resulting routing inside the clusters Q for each $v_Q \in P$ to obtain a path P' in graph G connecting the same pair of terminals as P. Since the set $\{P_e \mid e \in E(H)\}$ of paths causes a constant congestion in graph G from Property (C3), the resulting set of paths causes a constant congestion in G.

4.3 Proof of Theorem 14

We use the following lemma as a subroutine.

Lemma 5 *Let G' be any graph, and let $\mathcal{S}_1, \mathcal{S}_2$ be two sets of paths in G', where the paths in each set are edge-disjoint (but the paths in $\mathcal{S}_1 \cup \mathcal{S}_2$ may share edges). Assume further that all paths in $\mathcal{S}_1$ originate at the same vertex s. Then we can efficiently find a subset $\mathcal{S}_1' \subseteq \mathcal{S}_1$ of at least $|\mathcal{S}_1| - 2|\mathcal{S}_2|$ paths, and for each path $P \in \mathcal{S}_2$, another path $\tilde{P}$ connecting the same pair of vertices as P, such that, if we denote $\mathcal{S}_2' = \left\{\tilde{P} \mid P \in \mathcal{S}_2\right\}$, then:*

1. *All paths in $\mathcal{S}_1' \cup \mathcal{S}_2'$ are edge-disjoint.*
2. *Let E' and $\tilde{E}$ be the sets of edges used by at least one path in $\mathcal{S}_1 \cup \mathcal{S}_2$ and $\mathcal{S}_1' \cup \mathcal{S}_2'$ respectively. Then $\tilde{E} \subseteq E'$.*

In other words, the lemma re-routes the paths in $\mathcal{S}_2$, using the paths in $\mathcal{S}_1$, and then chooses $\mathcal{S}_1'$ to be the subset of paths in $\mathcal{S}_1$ that do not share edges with the new re-routed paths

in $\mathcal{S}_2'$. The rerouting guarantees that re-routed paths only overlap with at most $2|\mathcal{S}_2|$ paths in $\mathcal{S}_1$.

PROOF OF LEMMA 5. The proof is very similar to arguments used by Conforti et al. [14]. Given any pair (P, P') of paths, we say that paths P and P' intersect at edge e, if both paths contain edge e, and we say that P and P' intersect iff they share any edge.

We set up an instance of the stable matching problem in a multi-graph. In this problem, we are given a complete bipartite multigraph $G = (A, B, E)$, where $|A| = |B|$. Each vertex $v \in A \cup B$ specifies an ordering R_v of the edges adjacent to v in G. A complete matching M between the vertices of A and B is called stable iff, for every edge $e = (a, b) \in E \setminus M$, the following holds. Let e_a, e_b be the edges adjacent to a and b respectively in M. Then either a prefers e_a over e, or b prefers e_b over e. Conforti et al. [14], generalizing the famous theorem of Gale and Shapley [15], show an efficient algorithm to find a perfect stable matching M in any such multigraph.

Given the sets $\mathcal{S}_1, \mathcal{S}_2$ of paths, we set up an instance of the stable matching problem as follows. Set A contains a vertex $a(P)$ for each path $P \in \mathcal{S}_1$. For each path $P \in \mathcal{S}_2$, if x, y are the two endpoints of P, then we add two vertices $b(P, x)$ and $b(P, y)$ to B. In order to ensure that $|A| = |B|$, we add dummy vertices to B as needed.

For each pair $P \in \mathcal{S}_1$, $P' \in \mathcal{S}_2$ of paths, for each edge e that these paths share, we add two edges $(a(P), b(P', x))$ and $(a(P), b(P', y))$, where x and y are the endpoints of P', and we think of these new edges as representing the edge e. We add additional dummy edges as needed to turn the graph into a complete bipartite graph.

Finally, we define preference lists for vertices in A and B. For each vertex $a(P) \in A$, the edges incident to $a(P)$ are ordered according to the order in which they appear on path P, starting from s. The dummy edges incident to $a(P)$ are ordered arbitrarily at the end of the list.

Consider now some vertex $b(P, x) \in B$. We again order the edges incident to $b(P, x)$ according to the order in which their corresponding edges appear on the path P, when we traverse P starting from x. The dummy edges incident on $b(P, x)$ are added at the end of the list in an arbitrary order. The preference list of the vertex $b(P, y)$ is defined similarly, except that now we traverse P starting from y. Finally, the preference lists of the dummy vertices are arbitrary.

Let M be any perfect stable matching in the resulting graph. We let $\mathcal{S}_1' \subseteq \mathcal{S}_1$ be the subset of paths that are matched to the dummy vertices. Clearly, $|\mathcal{S}_1'| \geq |\mathcal{S}_1| - 2|\mathcal{S}_2|$. For each path $P \in \mathcal{S}_2$, we now define a path $\tilde{P}$, as follows. Let x, y be the two endpoints of P. If at least one of the vertices $b(P, x), b(P, y)$ participates in M via a dummy edge, then we let $\tilde{P} = P$, and we say that P is of type 1. Otherwise, let e, e' be the edges of M incident on $b(P, x)$ and $b(P, y)$, respectively, and let $P_1, P_2 \in \mathcal{S}_1$ be two paths such that $a(P_1)$ is the other endpoint of e and $a(P_2)$ is the other endpoint of e'. Let e_1, e_2 be the edges of the original graph that the edges e, e' represent. Let $\sigma_1(P)$ be the segment of P from x to e; $\sigma_2(P)$ the segment of P_1 from e to s; $\sigma_3(P)$ the segment of P_2 from s to e'; and $\sigma_4(P)$ the segment of P from e' to y. We set $\tilde{P}$ be the concatenation of $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$, and we say that P is of type 2. Let $\mathcal{S}_2' = \left\{\tilde{P} \mid P \in \mathcal{S}_2\right\}$. It now only remains to show that all paths in $\mathcal{S}_1' \cup \mathcal{S}_2'$ are edge-disjoint. It is immediate that the paths in $\mathcal{S}_1'$ are edge-disjoint, since the paths in $\mathcal{S}_1$ were edge-disjoint. We complete the proof in the following two claims.

Claim 2 *All paths in $\mathcal{S}_2'$ are edge-disjoint.*

PROOF. Assume otherwise, and let $\tilde{P}, \tilde{P}' \in \mathcal{S}_2'$ be any pair of paths that share an edge, say e. First, it is impossible that both P and P' are of type 1, since then $P, P' \in \mathcal{S}_2$, and so they must be edge-disjoint. So at least one of the two paths must be of type 2. Assume w.l.o.g. that it is P, and consider the four segments $\sigma_1(P), \sigma_2(P), \sigma_3(P), \sigma_4(P)$ of $\tilde{P}$, and the two edges e_1, e_2 that we have defined above. Let P_1 and P_2 be the paths in $\mathcal{S}_1$ on which the segments $\sigma_2(P), \sigma_3(P)$ lie.

If P' is of type 1, then it can only intersect $\sigma_2(P)$ or $\sigma_3(P)$, as the paths P and P' are edge-disjoint. Assume w.l.o.g. that P' intersects $\sigma_2(P)$, and let e' be any edge that they share. Let x' be the endpoint of P' such that the edge incident to $b(P', x')$ in M is a dummy edge. Then $b(P', x')$ prefers e' to its current matching, and $a(P_1)$ prefers e' to its current matching as well, as e' lies before e_1 on path P', a contradiction.

Assume now that P' is of type 2, and consider the segments $\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')$ of $\tilde{P}'$. Since the sets $\mathcal{S}_1, \mathcal{S}_2$ of paths are edge-disjoint, the only possibilities are that either one of the segments $\sigma_1(P)$, $\sigma_4(P)$ intersects one of the segments $\sigma_2(P')$, $\sigma_3(P')$, or one of the segments $\sigma_1(P')$, $\sigma_4(P')$ intersects one of the segments $\sigma_2(P)$, $\sigma_3(P)$. Assume w.l.o.g. that $\sigma_1(P)$ shares an edge e with $\sigma_2(P')$. Let x be the endpoint of P to which $\sigma_1(P)$ is adjacent, and let e_1 be the last edge on $\sigma_1(P)$, and let $P_1 \in \mathcal{S}_1$ be the path that shares e_1 with P. Then vertex $b(P, x)$ prefers the edge e to its current matching, as it appears before e_1 on P, starting from x. Similarly, $a(P_1)$ prefers e to its current matching, a contradiction. □

Claim 3 *Paths in $\mathcal{S}_1'$ and $\mathcal{S}_2'$ are edge-disjoint from each other.*

PROOF. Assume otherwise, and let $P \in \mathcal{S}_1'$, $P' \in \mathcal{S}_2'$ be two paths that share an edge e. It is impossible that P' is of type 1: otherwise, for some endpoint x of P', $b(P', x)$ is adjacent to a dummy edge in M, and $a(P)$ is also adjacent to a dummy edge in M, while both of them prefer e, a contradiction. Therefore, P' is of type 2. Consider the four segments $\sigma_1(P'), \sigma_2(P'), \sigma_3(P'), \sigma_4(P')$. Since the paths in each set $\mathcal{S}_1$ and $\mathcal{S}_2$ are edge-disjoint, the only possibility is that e belongs to either $\sigma_1(P')$, or to $\sigma_4(P')$. Assume w.l.o.g. that $e \in \sigma_1(P')$. Let x be the endpoint of P' to which $\sigma_1(P')$ is adjacent. Then $b(P', x)$ prefers e to its current matching, and similarly $a(P)$ prefers e to its current matching, a contradiction. □

We are now ready to complete the proof of Theorem 14. We build a graph G' from graph G, by replacing each edge of G with γ parallel edges. It is enough to define the subsets $\mathcal{P}_i^* \subseteq \mathcal{P}_i$, and the paths $\tilde{P}$ for $P \in \mathcal{P}''$ in graph G', such that, in the resulting set $\tilde{\mathcal{P}}' \cup \tilde{\mathcal{P}}''$, all paths are edge-disjoint. From now on we focus on finding these path sets in graph G'.

We perform k' iterations, where in iteration i we process the paths in set $\mathcal{P}_i$, and define a subset $\mathcal{P}_i^* \subseteq \mathcal{P}_i$. In each iteration, we may also change the paths in $\mathcal{P}''$, by replacing some of these paths with paths that have the same endpoints as the original paths (we call this process re-routing). We

maintain the invariant that at the beginning of iteration i, the paths in set $\mathcal{P}'' \cup \left(\bigcup_{i'=1}^{i-1} \mathcal{P}^*_{i'}\right)$ are edge-disjoint in G'. We now describe the ith iteration, for $1 \leq i \leq k'$.

Let $\mathcal{S}_1 = \mathcal{P}_i$, and let $\mathcal{S}_2$ be the collection of consecutive segments of paths in $\mathcal{P}''$ that are contained in S. The sets $\mathcal{S}_1$ and $\mathcal{S}_2$ of paths are both edge-disjoint in graph G', and so we can apply Lemma 5 to them, obtaining the sets $\mathcal{S}'_1$ and $\mathcal{S}'_2$ of paths. We then set $\mathcal{P}^*_i = \mathcal{S}'_1$, and modify every path $P'' \in \mathcal{P}''$ by removing the segments of $\mathcal{S}_2$ from it, and adding the corresponding segments of $\mathcal{S}'_2$ instead. Let $\mathcal{P}''$ be the collection of the resulting paths. Clearly, the paths in $\mathcal{P}''$ connect the same pairs of terminals as the original paths, and they continue to be edge-disjoint in G' (since the re-routing was only performed inside the graph $G'[S]$). Moreover, since the paths in all sets $\mathcal{P}_1, \ldots, \mathcal{P}_i$ are edge-disjoint, and we have only used the edges of the paths in $\mathcal{P}_i$ to re-route the paths in $\mathcal{P}''$, the paths in set $\mathcal{P}'' \cup \left(\bigcup_{i'=1}^{i} \mathcal{P}^*_{i'}\right)$ are edge-disjoint in G'. Finally, observe that $|\mathcal{S}_2| \leq \gamma L \log n$, since $|\operatorname{out}_G(S)| \leq L \log n$, and every path in $\mathcal{S}_2$ contains at least one edge of $\operatorname{out}_{G'}(S)$. Therefore, $|\mathcal{P}^*_i| \geq |\mathcal{P}_i| - 2L\gamma \log n$. Once we process all sets $\mathcal{P}_1, \ldots \mathcal{P}_{k'}$, our output is $\{\mathcal{P}^*_i\}_{i=1}^{k'}$, and we output the final set $\mathcal{P}''$ that contains a re-routed path $\tilde{P}$ for each path P in the original set.

5. HARDNESS OF BASIC-ICF

In this section we prove Theorem 2, by performing a simple reduction from the Congestion Minimization problem. We use the following theorem, due to Andrews and Zhang [4].

Theorem 16 *Let G be any n-vertex graph with unit edge capacities, and let $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ be a collection of source-sink pairs. Then, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$, no efficient algorithm can distinguish between the following two cases:*

- *(*YES-INSTANCE*): There is a collection $\mathcal{P}$ of paths that causes congestion 1, and for each $1 \leq i \leq k$, there is a path connecting s_i to t_i in $\mathcal{P}$.*
- *(*NO-INSTANCE*): For any collection $\mathcal{P}$ of paths that contains, for each $1 \leq i \leq k$, a path connecting s_i to t_i, the congestion due to $\mathcal{P}$ is at least $\Omega(\log\log n/\log\log\log n)$.*

Let G be any n-vertex graph with unit edge capacities and a set $\mathcal{M} = \{(s_1, t_1), \ldots, (s_k, t_k)\}$ of source-sink pairs. We build a new graph G', where we replace every edge in G by D parallel edges. Observe that if G is a YES-INSTANCE, then there is a collection $\mathcal{P}$ of paths that causes congestion 1 in G, and every demand pair $(s_i, t_i) \in \mathcal{M}$ is connected by a path in $\mathcal{P}$. We then immediately obtain a collection $\mathcal{P}'$ of paths in graph G' that causes congestion 1, and every demand pair (s_i, t_i) is connected by D paths, by simply taking D copies of each path in $\mathcal{P}$. Assume now that G is a NO-INSTANCE, and assume for contradiction that there is a collection $\mathcal{P}'$ of paths in graph G' that causes congestion at most c, and every demand pair (s_i, t_i) is connected by at least one path in $\mathcal{P}'$. Then set $\mathcal{P}'$ of paths defines a collection $\mathcal{P}$ of paths in graph G, that causes congestion at most Dc, and every demand pair (s_i, t_i) is connected by at least one path in $\mathcal{P}$.

From Theorem 16, no efficient algorithm can distinguish between the case where there is a collection $\mathcal{P}'$ of edge-disjoint paths in G', where every demand pair is connected by D paths, and the case where any collection $\mathcal{P}'$ of paths, containing at least one path connecting every demand pair causes congestion c, for any values D and c with $Dc = O(\log\log n/\log\log\log n)$, unless $\mathsf{NP} \subseteq \mathsf{ZPTIME}(n^{\mathrm{poly}\log n})$.

6. HARDNESS OF GROUP-ICF

The goal of this section is to prove Theorem 3. We start by introducing a new scheduling problem, called Max-Min Interval Scheduling (MMIS), and show that it can be cast as a special case of group-ICF. We show later that MMIS is hard to approximate.

6.1 Max-Min Interval Scheduling

In the MMIS problem, we are given a collection $\mathcal{J} = \{1, \ldots, n\}$ of n jobs, and for each job j, we are given a set $\mathcal{I}_j$ of **disjoint** closed intervals on the time line. Given any set $\mathcal{I}$ of time intervals, the *congestion* of $\mathcal{I}$ is the maximum, over all time points t, of the number of intervals in $\mathcal{I}$ containing t. Speaking in terms of scheduling, this is the number of machines needed to execute the jobs during the time intervals in $\mathcal{I}$. The goal of MMIS is to find, for each job j, a subset $\mathcal{I}^*_j \subseteq \mathcal{I}_j$ of intervals such that, if we denote $\mathcal{I}^* = \bigcup_{j=1}^{n} \mathcal{I}^*_j$, then the intervals in $\mathcal{I}^*$ are disjoint (or equivalently cause congestion 1). The value of the solution is the minimum, over all jobs $j \in \mathcal{J}$, of $|\mathcal{I}^*_j|$. Given an instance $\mathcal{J}$ of MMIS, we denote by $N(\mathcal{J})$ the total number of intervals in $\bigcup_{j\in\mathcal{J}} \mathcal{I}_j$. In the following theorem, we relate MMIS to group-ICF.

Theorem 17 *Given any instance $\mathcal{J}$ of MMIS, we can efficiently construct an instance $(G, \mathcal{D})$ of group-ICF on a line graph, such that $|V(G)| \leq 2N(\mathcal{J})$ and the following holds:*

- *If OPT is the value of the optimal solution to $\mathcal{J}$ with congestion 1, then there is a collection $\mathcal{P}^*$ of edge-disjoint paths in G, where each pair (S_i, T_i) is connected by OPT paths in $\mathcal{P}^*$, and*
- *Given any set $\mathcal{P}^*$ of paths in G, where every pair (S_i, T_i) is connected by at least D' paths, and the total congestion cased by $\mathcal{P}^*$ is bounded by c, we can efficiently find a solution $\mathcal{I}^*$ to instance $\mathcal{J}$ of value D' and congestion c.*

PROOF. Given an instance of the MMIS problem, we construct an instance of group-ICF on a line graph as follows. Let $\mathcal{J} = \{1, \ldots, n\}$ be the input set of jobs, and let $\mathcal{I} = \bigcup_{j=1}^{n} \mathcal{I}_j$. Let $\mathcal{S}$ be the set of endpoints of all intervals in $\mathcal{I}$. We create a line graph $G = (V, E)$, whose vertex set is $\mathcal{S}$, and the vertices are connected in the order in which they appear on the time line. Clearly, $|V(G)| \leq 2N(\mathcal{J})$. For each job j, we create a demand pair (S_j, T_j), as follows. Assume w.l.o.g. that $\mathcal{I}_j = \{I_1, I_2, \ldots, I_r\}$, where the intervals are ordered left-to-right (since all intervals in $\mathcal{I}_j$ are disjoint, this order is well-defined). For each $1 \leq x \leq r$, if x is odd, then we add the left endpoint of I_x to S_j and its right endpoint to T_j; otherwise, we add the right endpoint to S_j and the left endpoint to T_j. In the end, $|S_j| = |T_j| = |\mathcal{I}_j|$. This concludes the definition of the group-ICF instance. Let OPT be the value of the optimal solution to the MMIS instance (with no congestion), and let $\mathcal{I}^* = \bigcup_{j\in\mathcal{J}} \mathcal{I}^*_j$ be the optimal solution to the MMIS problem instance. Notice that for each job $j \in \mathcal{J}$, for each interval $I \in \mathcal{I}^*_j$, one of its endpoints is in S_j and the other is in T_j. For each interval $I \in \mathcal{I}^*$, we

add the path connecting the endpoints of I to our solution. It is immediate to see that each group (S_i, T_i) is connected by OPT paths, and since intervals in $\mathcal{I}^*$ are disjoint, the congestion is bounded by 1.

Assume now that we are given a solution $\mathcal{P}^* = \bigcup_{j=1}^{n} \mathcal{P}_j^*$ to the group-ICF instance, where $\mathcal{P}_j^*$ is the set of paths connecting S_j to T_j, such that $|\mathcal{P}_j^*| \geq D'$, and the paths in $\mathcal{P}^*$ cause congestion at most c. We now transform $\mathcal{P}^*$ into a solution of value D' and congestion c for the original MMIS instance, as follows.

Consider any path P, whose endpoints are x, y, where one of these two vertices belongs to S_j and the other to T_j. We say that P is *canonical* iff x, y are endpoints of some interval $I \in \mathcal{I}_j$. If some path $P \in \mathcal{P}_j^*$ is not canonical, we can transform it into a canonical path, without increasing the overall congestion, as follows. We claim that path P must contain some interval $I \in \mathcal{I}_j$. Indeed, otherwise, let $\mathcal{S}_j$ be the set of endpoints of the intervals in $\mathcal{I}_j$. Then x and y are two consecutive vertices in $\mathcal{S}_j$, they are not endpoints of the same interval in $\mathcal{I}_j$, and yet one of them belongs to S_j and the other to T_j. But the sets S_j and T_j were defined in a way that makes this impossible. Therefore, P contains some interval $I \in \mathcal{I}_j$. We truncate path P so that it starts at the left endpoint of I and ends at the right endpoint of I. Once we process all paths in $\mathcal{P}^*$, we obtain a solution to the group-ICF problem instance, where all paths are canonical and the congestion is still bounded by c. This solution immediately gives us a solution of value D' and congestion at most c to the MMIS problem instance. □

We note that a scheduling problem closely related to MMIS is Machine Minimization Job Scheduling, where the goal is to select one time interval for every job, so as to minimize the total congestion. This problem admits an $O(\log n / \log\log n)$-approximation via the Randomized LP-Rounding technique of Raghavan and Thompson [23]. Chuzhoy and Naor [13] have shown that it is $\Omega(\log\log n)$-hard to approximate.

In the following theorem, we prove hardness of the MMIS problem, which, combined with Theorem 17, immediately implies Theorem 3. Its proof very closely follows the hardness of approximation proof of [13] for Machine Minimization Job Scheduling and appears in the full version of the paper.

Theorem 18 *Suppose we are given an instance $\mathcal{J}$ of* MMIS*, and let $N = N(\mathcal{J})$. Let c be any integer, $0 < c \leq O(\log\log N)$ and let $D = O\left(N^{1/(2^{2c+3})}\right)$. Then no efficient algorithm can distinguish between the following two cases:*

- *(*Yes-Instance*): the value of the optimal solution with congestion 1 is at least D, and*
- *(*No-Instance*): any solution $\mathcal{I}^*$, where for all $j \in J$, $|\mathcal{I}_j^*| \geq 1$, causes congestion at least c.*

unless NP $\subseteq$ DTIME$(n^{O(\log\log n)})$.

7. AN ALGORITHM FOR GROUP-ICF

We again start with a special case of the problem, where all edge capacities are unit, and all demands are uniform. By appropriately scaling the demands, we can then assume w.l.o.g. that $\lambda_{\text{OPT}} = 1$, and the demand for each pair (S_i, T_i) is D. Let $m = \lceil 16 \log n \rceil$ be a parameter. We later define a parameter $\Delta = k \operatorname{poly}\log n$, and we assume throughout the algorithm that:

$$D \geq 640 \Delta m \alpha_{\text{EDP}} \ln^2 n = k \operatorname{poly}\log n, \tag{1}$$

by setting $\Delta' = 640 \Delta m \alpha_{\text{EDP}} \ln^2 n$.

We say that a group-ICF instance $(G, \mathcal{D})$ is a *canonical instance* iff for all $1 \leq i \leq k$, $S_i = \left\{s_1^i, \ldots, s_{mD}^i\right\}$, $T_i = \left\{t_1^i, \ldots, t_{mD}^i\right\}$, and there is a set of paths

$$\mathcal{P} = \left\{P_j^i \mid 1 \leq i \leq k, 1 \leq j \leq mD\right\}$$

where path P_j^i connects s_j^i to t_j^i, and the paths in $\mathcal{P}$ cause congestion at most $2m$ in G. We denote by $\mathcal{M}_i = \left\{(s_j^i, t_j^i)\right\}_{j=1}^{mD}$ the set of pairs corresponding to (S_i, T_i) for each $1 \leq i \leq k$, and we associate a canonical fractional solution f^* with such an instance, where we send $1/(2m)$ flow units along each path P_j^i. The value of such a solution is $D/2$ - the total amount of flow sent between each pair (S_i, T_i). Let $\mathcal{M} = \bigcup_{i=1}^{k} \mathcal{M}_i$. The following lemma allows us to assume w.l.o.g. that the input instance is canonical. The proof uses standard randomized rounding and appears in the full version.

Lemma 6 *Given any instance $(G, \mathcal{D})$ of* group-ICF *with unit edge capacities and uniform demands $D_i = D$ for all $1 \leq i \leq k$, where the value of the optimal solution $\lambda_{OPT} = 1$, we can efficiently compute a canonical instance $(G, \mathcal{D}', \mathcal{P})$, where for each $1 \leq i \leq k$, $|\mathcal{P}_i| = Dm$. Moreover, any integral solution to the canonical instance gives an integral solution of the same value and congestion to the original instance.*

From now on, we will assume that the input instance is a canonical one, and that the set $\mathcal{P}$ of paths is given. Throughout this section, the parameter L in the definition of small and critical clusters is set to $L = O(\log^{25} n)$, and we set the precise value of L later.

7.1 Split Instances and Good Q-J Decompositions

In this section we introduce two special cases of the group-ICF problem and show efficient algorithms for solving them. In the following section we show an algorithm for the general problem, which decomposes an input instance of group-ICF into several sub-instances, each of which belongs to one of the two special cases described here.

Split Instances.

The first special case that we define is a split instance. Suppose we are given a canonical instance $(G, \mathcal{D})$ of the group-ICF problem, with the corresponding set $\mathcal{P}$ of paths connecting the demand pairs in $\bigcup_{i=1}^{k} \mathcal{M}_i$. Assume further that we are given a collection $\mathcal{C} = \{C_1, \ldots, C_\ell\}$ of disjoint vertex subsets of G, such that each path $P \in \mathcal{P}$ is completely contained in one of the sets C_h. For each $1 \leq i \leq k$, for each $1 \leq h \leq \ell$, let $\mathcal{P}_i(C_h) \subseteq \mathcal{P}_i$ be the subset of paths contained in C_h. We say that instance $(G, \mathcal{D})$ is a split instance iff for each $1 \leq i \leq k$, for each $1 \leq h \leq \ell$, $|\mathcal{P}_i(C_h)| \leq \frac{D}{4\alpha_{\text{EDP}} \cdot \ln^2 n} = \frac{D}{\operatorname{poly}\log n}$.

Theorem 19 *Let $(G, \mathcal{D})$ be a canonical split instance as described above. Then there is an efficient randomized al-*

gorithm that finds a collection $\mathcal{R}$ of paths that cause a congestion of at most η_{EDP} in G, and for each $1 \leq i \leq k$, at least $\frac{D}{64\alpha_{\text{EDP}} \cdot \ln^2 n} = \frac{D}{\text{poly} \log n}$ paths connect the vertices of S_i to the vertices of T_i in $\mathcal{R}$ w.h.p.

Proof. For each $1 \leq h \leq \ell$, let $\mathcal{P}(C_h) \subseteq \mathcal{P}$ be the subset of paths contained in C_h, and let $\mathcal{M}(C_h) \subseteq \mathcal{M}$ be the set of pairs of terminals that these paths connect. Recall that the paths in set $\mathcal{P}(C_h)$ cause a congestion of at most $2m$ in graph $G[C_h]$. Therefore, if we route $1/(2m)$ flow units along each path $P \in \mathcal{P}(C_h)$, then we obtain a feasible fractional solution to the EDP instance on graph $G[C_h]$ and the set $\mathcal{M}(C_h)$ of demands, where the value of the solution is $\frac{|\mathcal{M}(C_h)|}{2m}$.

Let $N = 2m\alpha_{\text{EDP}} \cdot \ln n$. We partition the set $\mathcal{M}(C_h)$ into N subsets $\mathcal{M}_1(C_h), \ldots, \mathcal{M}_N(C_h)$, and for each $1 \leq z \leq N$, we find a collection $\mathcal{R}_z(C_h)$ of paths contained in graph $G[C_h]$ that connect the demands in $\mathcal{M}_z(C_h)$ and cause congestion at most η_{EDP}.

We start with the set $\mathcal{M}(C_h)$ of demands, and apply Theorem 5 to input $(G[C_h], \mathcal{M}(C_h))$. Since there is a fractional solution of value $\frac{|\mathcal{M}(C_h)|}{2m}$, we obtain a collection $\mathcal{R}_1(C_h)$ of paths connecting a subset $\mathcal{M}_1(C_h) \subseteq \mathcal{M}(C_h)$ of at least $\frac{|\mathcal{M}(C_h)|}{2m\alpha_{\text{EDP}}}$ demand pairs. We then remove the pairs in $\mathcal{M}_1(C_h)$ from $\mathcal{M}(C_h)$ and continue to the next iteration. Clearly, after N iterations, every pair in the original set $\mathcal{M}(C_h)$ belongs to one of the subsets $\mathcal{M}_1(C_h), \ldots, \mathcal{M}_N(C_h)$. For each such subset $\mathcal{M}_z(C_h)$, we have computed an integral routing $\mathcal{R}_z(C_h)$ of the pairs in $\mathcal{M}_z(C_h)$ inside $G[C_h]$, with congestion at most η_{EDP}. We now choose an index $z \in \{1, \ldots, N\}$ uniformly at random, and set $\mathcal{M}'(C_h) = \mathcal{M}_z(C_h)$, and $\mathcal{R}(C_h) = \mathcal{R}_z(C_h)$. We view $\mathcal{R}(C_h)$ as the final routing of pairs in $\mathcal{M}'(C_h)$ inside the cluster C_h, and we say that all pairs in $\mathcal{M}'(C_h)$ are routed by this solution. Notice that the probability that a pair in $\mathcal{M}(C_h)$ is routed is $1/N$. The output of the algorithm is $\mathcal{R} = \bigcup_{h=1}^{\ell} \mathcal{R}(C_h)$.

We say that a demand pair (S_i, T_i) is satisfied iff $\mathcal{R}$ contains at least $\frac{mD}{2N} = \frac{D}{4\alpha_{\text{EDP}} \cdot \ln n}$ paths connecting the vertices of S_i to the vertices of T_i. We now show that w.h.p. every demand pair (S_i, T_i) is satisfied.

Indeed, consider some demand pair (S_i, T_i). Recall that $|\mathcal{P}_i| = Dm$, and for each cluster $C_h \in \mathcal{C}$, $\mathcal{P}_i(C_h) \subseteq \mathcal{P}_i$ is the subset of paths contained in C_h. Let $\mathcal{M}_i(C_h) \subseteq \mathcal{M}_i$ be the set of pairs of endpoints of paths in $\mathcal{P}_i(C_h)$. Denote $D^* = \frac{D}{64\alpha_{\text{EDP}} \ln^2 n}$, and recall that we $|\mathcal{M}_i(C_h)| \leq D^*$ must hold. We now define a random variable $y_{i,h}$ as follows. Let $n_{i,h}$ be the number of pairs of vertices in $\mathcal{M}_i(C_h)$ that are routed by $\mathcal{R}(C_h)$. Then $y_{i,h} = \frac{n_{i,h}}{D^*}$. Observe that the variables $\{y_{i,h}\}_{h=1}^{\ell}$ are independent random variables that take values in $[0, 1]$. Let $Y_i = \sum_{h=1}^{\ell} y_{i,h}$. Then the expectation of Y_i is $\mu_i = \frac{mD}{D^* \cdot N} = \frac{64m\alpha_{\text{EDP}} \ln^2 n}{2m\alpha_{\text{EDP}} \cdot \ln n} = 32 \ln n$.

The probability that (S_i, T_i) is not satisfied is the probability that $Y_i < \frac{mD}{2D^* \cdot N} = \mu_i/2$. By standard Chernoff bounds, this is bounded by $e^{-\mu_i/8} \leq e^{-4 \ln n} = 1/n^4$. From the union bound, with probability at least $(1 - 1/n^3)$, all pairs (S_i, T_i) are satisfied. □

Good Q-J Decompositions.

Suppose we are given a canonical instance $(G, \mathcal{D})$ of the group-ICF problem, and any valid Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of the graph G. Recall that for each critical cluster $Q \in \mathcal{Q}$, we are given a partition $\pi(Q)$ of Q into small clusters that have the bandwidth property. Let $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$. We say that the Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ is *good* iff $\mathcal{Q} \neq \emptyset$, and no demand pair (s_j^i, t_j^i) is contained in a single cluster in $\mathcal{X}$. In other words, for each $1 \leq i \leq k$, for each $1 \leq j \leq mD$, vertices s_j^i and t_j^i must belong to distinct clusters in $\mathcal{X}$. We show that if we are given a good Q-J decomposition for G, then we can efficiently find a good integral solution for it.

For technical reasons that will become apparent later, we state the next theorem for canonical instances with non-uniform demands.

Theorem 20 *Assume that we are given a graph G, and for $1 \leq i \leq k$, two collections $S_i = \left\{s_1^i, \ldots, s_{D_i}^i\right\}$, $T_i = \left\{t_1^i, \ldots, t_{D_i}^i\right\}$ of vertices, where $D_i \geq \Omega\left(L^2 \log^{11} n\right)$. Assume further that we are given a set of paths,*

$$\mathcal{P} = \left\{P_j^i \mid 1 \leq i \leq k, 1 \leq j \leq D_i\right\},$$

where path P_j^i connects s_j^i to t_j^i, and the paths in $\mathcal{P}$ cause congestion at most $2m$ in G. Assume also that we are given a good Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ for G. Then there is an efficient algorithm that finds an integral solution to the group-ICF instance, whose congestion is at most c_{good}, and for each $1 \leq i \leq k$, at least $\lfloor D_i/\alpha_{\mathsf{good}} \rfloor$ paths connect vertices of S_i to vertices of T_i. Here, $\alpha_{\mathsf{good}} = O(\log^{60} n \operatorname{poly} \log \log n)$, and c_{good} is a constant.

Proof. Recall that we are given a canonical fractional solution, with a collection $\mathcal{P} = \left\{P_j^i : 1 \leq i \leq k, 1 \leq j \leq D_i\right\}$ of paths, such that path P_j^i connects s_j^i to t_j^i, and no path in $\mathcal{P}$ has its two endpoints in the same cluster $X \in \mathcal{X}$, where $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$. We view each path $P_j^i \in \mathcal{P}$ as starting at vertex s_j^i and terminating at t_j^i. We will repeatedly use the following claim, whose proof follows from standard Chernoff bound.

Claim 4 *Let $\mathcal{P}'$ be any collection of paths in G, and let $(\mathcal{P}_1', \ldots, \mathcal{P}_k')$ be any partition of $\mathcal{P}'$. Assume that we are given another partition $\mathcal{G}$ of the set $\mathcal{P}'$ of paths into groups of size at most q each, and assume further that for each $1 \leq i \leq k$, $|\mathcal{P}_i'| \geq 32q \log n$. Let $\mathcal{P}'' \subseteq \mathcal{P}'$ be a subset of paths, obtained by independently selecting, for each group $U \in \mathcal{G}$, a path $P_U \in U$ uniformly at random, so $\mathcal{P}'' = \{P_U \mid U \in \mathcal{G}\}$. Then for each $1 \leq i \leq k$, $|\mathcal{P}_i' \cap \mathcal{P}''| \geq \frac{|\mathcal{P}_i'|}{2q}$ with high probability.*

We say that a path $P \in \mathcal{P}$ is *critical* iff it is contained in some critical cluster $Q \in \mathcal{Q}$. We say that a pair (S_i, T_i) is critical iff the number of paths in $\mathcal{P}_i$ that are critical is at least $|\mathcal{P}_i|/2$. Otherwise, we say that (S_i, T_i) is a *regular* pair. We first show how to route the critical pairs, and then show an algorithm for routing regular pairs.

Routing Critical Pairs.

Let $\tilde{\mathcal{P}} \subseteq \mathcal{P}$ be the set of all critical paths, and let $\mathcal{X}^Q = \bigcup_{Q \in \mathcal{Q}} \pi(Q)$. For each cluster $X \in \mathcal{X}^Q$, we define two sets of paths: $U_1(X)$, containing all paths in $\tilde{\mathcal{P}}$ whose first endpoint belongs to X, and $U_2(X)$, containing all paths in $\tilde{\mathcal{P}}$ whose last endpoint belongs to X. Since cluster X cannot contain both endpoints of any path in $\tilde{\mathcal{P}}$, and the paths in $\tilde{\mathcal{P}}$ cause congestion at most $2m$, while $|\text{out}(X)| \leq L$, we

have that $|U_1(X)|, |U_2(X)| \leq 2mL$ for all $X \in \mathcal{X}^Q$. For each cluster $X \in \mathcal{X}^Q$, we then select, uniformly independently at random, one path $P_1(X) \in U_1(X)$, and one path $P_2(X) \in U_2(X)$. We then let $\tilde{\mathcal{P}}' \subseteq \tilde{\mathcal{P}}$ be the subset of paths P that were selected twice in this process. That is, $\tilde{\mathcal{P}}' = \left\{P_1(X) \mid X \in \mathcal{X}^Q\right\} \cap \left\{P_2(X) \mid X \in \mathcal{X}^Q\right\}$. Notice that both $\{U_1(X)\}_{X \in \mathcal{X}^Q}$ and $\{U_2(X)\}_{X \in \mathcal{X}^Q}$ are partitions of $\tilde{\mathcal{P}}$ into subsets of size at most $2mL$. Therefore, from Claim 4, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \tilde{\mathcal{P}}'| \geq \frac{|\mathcal{P}_i|}{16m^2L^2}$ w.h.p.

We now fix some critical cluster $Q \in \mathcal{Q}$. Let $\mathcal{P}_Q \subseteq \tilde{\mathcal{P}}'$ denote the subset of paths in $\tilde{\mathcal{P}}'$ that are contained in Q, let $\mathcal{M}_Q$ be the set of pairs of their endpoints, and $\mathcal{T}_Q$ the subset of terminals that serve as endpoints to the paths in $\mathcal{P}_Q$. Recall that each small cluster $C \in \pi(Q)$ contains at most two terminals from $\mathcal{T}'$. We augment the graph G, by adding, for every terminal $t \in \mathcal{T}_Q$, an edge e_t connecting t to a new vertex t'. Let G_Q denote this new graph (but note that the new vertices t' do not belong to Q). We now show that Q still has the weight property in this new graph, and each cluster $C \in \pi(Q)$ still has the bandwidth property (with slighter weaker parameters). We can then use Theorem 9 for routing on critical clusters, to route the pairs in $\mathcal{M}_Q$. The proof of the following claim appears in the full version of the paper.

Claim 5 *Each cluster $C \in \pi(Q)$ is $(\alpha_S/3)$-well-linked in graph G_Q, and $(Q, \pi(Q))$ has the weight property with parameter $\lambda' = \lambda/3$.*

We can now use Theorem 9 to find a grouping $\mathcal{G}_Q$ of the terminals in $\mathcal{T}_Q$ into groups of size $O(Z) = O(\log^4 n)$, such that for any set D of $(1, \mathcal{G}_Q)$-restricted demands, there is an efficient randomized algorithm that w.h.p. routes D integrally in $G[Q]$ with constant congestion. Grouping $\mathcal{G}_Q$ defines two partitions $\mathcal{G}_Q^1, \mathcal{G}_Q^2$ of the paths in $\mathcal{P}_Q$, as follows: for each group $U \in \mathcal{G}_Q$, we have a subset $\mathcal{P}_1(U) \subseteq \mathcal{P}_Q$ of paths whose first endpoint belongs to U, and a subset $\mathcal{P}_2(U) \subseteq \mathcal{P}_Q$ of paths whose last endpoint belongs to U. We let $\mathcal{G}_Q^1 = \{\mathcal{P}_1(U) \mid U \in \mathcal{G}_Q\}$, and $\mathcal{G}_Q^2 = \{\mathcal{P}_2(U) \mid U \in \mathcal{G}_Q\}$. For each group $U \in \mathcal{G}$, we randomly sample one path in $\mathcal{P}_1(U)$ and one path in $\mathcal{P}_2(U)$. We let $\mathcal{P}'_Q$ be the set of all paths that have been selected twice, once via their first endpoint and once via their last endpoint, and we let $\tilde{\mathcal{P}}'' = \bigcup_{Q \in \mathcal{Q}} \mathcal{P}'_Q$. From Claim 4, for each critical (S_i, T_i) pair, $|\mathcal{P}_i \cap \tilde{\mathcal{P}}''| \geq \Omega\left(\frac{|\mathcal{P}_i|}{m^2L^2Z^2}\right) = \Omega\left(\frac{|\mathcal{P}_i|}{L^2 \log^{10} n}\right)$ w.h.p. For each $Q \in \mathcal{Q}$, let $\mathcal{M}'_Q$ be the set of pairs of endpoints of paths in $\mathcal{P}'_Q$. Then $\mathcal{M}'_Q$ defines a set of $(2, \mathcal{G}_Q)$-restricted demands on the set $\mathcal{T}_Q$ of terminals, and from Theorem 9, there is a randomized algorithm to route these demands in $G[Q]$ with constant congestion.

Routing Regular Pairs.

We use the graph H, given by Theorem 11. The algorithm consists of two steps. In the first step, we route some of the terminals to the boundaries of the critical clusters, and create what we call "fake terminals". This step is very similar to the algorithm of Andrews [1]. In this way, we transform the problem of routing the original terminal pairs in graph G into a problem of routing the new fake terminal pairs in graph H. The second step is very similar to the proof of Theorem 13: we split graph H into $x = \text{poly} \log n$ subgraphs $H_1, \ldots, H_x$ using standard edge sampling, and route a subset of the fake demand pairs in each graph H_j using the algorithm of Rao and Zhou [24].

Since we now focus on regular pairs only, to simplify notation, we assume that we are given a collection $\{(S_i, T_i)\}_{i=1}^k$ of regular demand pairs, and a collection $\mathcal{P} = \left\{P_j^i\right\}_{\substack{1 \leq i \leq k,\\ 1 \leq j \leq D'_i}}$ of paths, where $D'_i = D_i/2$, such that path P_j^i connects s_j^i to t_j^i. We assume that all paths P_j^i are regular. For each $1 \leq i \leq k$, $S_i = \left\{s_1^i, \ldots, s_{D'_i}^i\right\}$, and $T_i = \left\{t_1^i, \ldots, t_{D'_i}^i\right\}$. Let $\mathcal{T} = \bigcup_{i=1}^k (S_i \cup T_i)$ be the set of all terminals, and for $1 \leq i \leq k$, let $\mathcal{M}_i = \left\{(s_j^i, t_j^i)\right\}_{j=1}^{D'_i}$ be the set of the pairs of endpoints of paths in set $\mathcal{P}_i = \left\{P_1^i, \ldots, P_{D_i}^i\right\}$. Denote by $\mathcal{T}^J$ and $\mathcal{T}^Q$ the sets of all terminals contained in the J- and the Q-clusters respectively, that is, $\mathcal{T}^J = \mathcal{T} \cap \left(\bigcup_{C \in \mathcal{J}} C\right)$, and $\mathcal{T}^Q = \mathcal{T} \cap \left(\bigcup_{C \in \mathcal{Q}} C\right)$.

Step 1: Defining fake terminal pairs.

The goal of this step is to define new pairs of terminals, that we call fake terminal pairs, in graph H, so that a good routing of the fake terminal pairs in graph H immediately translates into a good routing of the original pairs in graph G. This step largely follows the ideas of [1]. Let $E^Q = \bigcup_{Q \in \mathcal{Q}} \text{out}(Q)$. Our first step is to route all terminals in $\mathcal{T}$ to the edges of E^Q. This is done using the following three lemmas, whose proofs appear in the full version of the paper.

Lemma 7 *There is an efficient algorithm to find a partition $\mathcal{G}^J$ of the set $\mathcal{T}^J$ of terminals into groups of size at most $48m$, such that for any $(2, \mathcal{G}^J)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}^J$ of terminals, there is an efficient algorithm to find a set $\mathcal{P}_J : \mathcal{T}' \rightsquigarrow_{12} E^Q$ of paths in G.*

Lemma 8 *We can efficiently find a partition $\mathcal{G}^Q$ of the set $\mathcal{T}^Q$ of terminals into groups of size at most $48m$, such that for any $(2, \mathcal{G}^Q)$-restricted subset $\mathcal{T}' \subseteq \mathcal{T}^Q$ of terminals, there is an efficient algorithm to find a set $\mathcal{P}_Q : \mathcal{T}' \rightsquigarrow_{12} E^Q$ of paths G.*

Let $\mathcal{G} = \mathcal{G}^Q \cup \mathcal{G}^J$ be the partition of terminals in $\mathcal{T}$ obtained from Lemmas 7 and 8. For each group $U \in \mathcal{G}$ of terminals, we define two subsets of paths: $\mathcal{P}_1(U) \subseteq \mathcal{P}$ contains all paths whose first endpoint belongs to U, and $\mathcal{P}_2(U) \subseteq \mathcal{P}$ contains all paths whose last endpoint belongs to U. We then select, independently uniformly at random, two paths $P_1(U) \in \mathcal{P}_1(U)$ and $P_2(U) \in \mathcal{P}_2(U)$. Let $\mathcal{P}' \subseteq \mathcal{P}$ be the subset of paths that have been selected twice, that is, $\mathcal{P}' = \{P_1(U) \mid U \in \mathcal{G}\} \cap \{P_2(U) \mid U \in \mathcal{G}\}$. Since both $\{\mathcal{P}_1(U)\}_{U \in \mathcal{G}}$ and $\{\mathcal{P}_2(U)\}_{U \in \mathcal{G}}$ define partitions of the paths in $\mathcal{P}$ into sets of size at most $48m$, from Claim 4, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \mathcal{P}'| \geq \frac{|\mathcal{P}_i|}{4608m^2}$ w.h.p.

Let $\mathcal{T}' \subseteq \mathcal{T}$ be the set of terminals that serve as endpoints for paths in $\mathcal{P}'$. Since set $\mathcal{T}'$ is $(2, \mathcal{G})$-restricted, from Lemmas 7 and 8, there is a collection $\mathcal{R} : \mathcal{T}' \rightsquigarrow_{24} E^Q$ of paths in graph G. For each terminal $t \in \mathcal{T}'$, set $\mathcal{R}$ contains a path P_t, connecting t to some edge $e_t \in E^Q$. We define a mapping $f : \mathcal{T}' \to E^Q$, where $f(t) = e_t$.

Recall that for each critical cluster $Q \in \mathcal{Q}$, Theorem 9 gives a partition $\mathcal{G}(Q)$ of the edges of out(Q) into subsets of size at most $3Z = O(\log^4 n)$.

Consider some edge $e \in E^Q$. If there is a single critical cluster $Q \in \mathcal{Q}$ such that $e \in \text{out}(Q)$, then we say that

e belongs to Q. Otherwise, if there are two such clusters $Q_1, Q_2 \in \mathcal{Q}$, then we select one of them arbitrarily, say Q_1, and we say that e belongs to Q_1. We will view $\mathcal{G}(Q)$ as a partition of only those edges in $\text{out}(Q)$ which belong to Q, and we will ignore all other edges. Let $\mathcal{G}' = \bigcup_{Q\in\mathcal{Q}} \mathcal{G}(Q)$, so $\mathcal{G}'$ is a partition of E^Q. Our final step is to sample the paths in $\mathcal{P}'$, such that for each group $U \in \mathcal{G}'$, there are at most two paths whose endpoints are mapped to the edges of U.

For each group $U \in \mathcal{G}'$, we define a subset $\mathcal{P}_1(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose first endpoint t is mapped to an edge of U, that is, $f(t) \in U$, and similarly, a subset $\mathcal{P}_2(U) \subseteq \mathcal{P}'$ of paths, containing all paths whose last endpoint is mapped to an edge of U. We then select, uniformly independently at random, a path $P_1(U) \in \mathcal{P}_1(U)$, and a path $P_2(U) \in \mathcal{P}_2(U)$, and let $\mathcal{P}'' \subseteq \mathcal{P}'$ be the subset of paths that have been selected twice, that is, $\{P_1(U) \mid U \in \mathcal{G}\} \cap \{P_2(U) \mid U \in \mathcal{G}\}$. Since each set $|\mathcal{P}_1(U)|, |\mathcal{P}_2(U)| \leq 3Z = O(\log^4 n)$, from Claim 4, for each $1 \leq i \leq k$, $|\mathcal{P}_i \cap \mathcal{P}''| \geq \Omega\left(\frac{D_i}{m^2 Z^2}\right) = \Omega\left(\frac{D_i}{\log^{10} n}\right)$ w.h.p.

Let $\mathcal{T}'' \subseteq \mathcal{T}$ be the set of terminals that serve as endpoints of paths in $\mathcal{P}''$, and let $\mathcal{R}'' \subseteq \mathcal{R}$ be their corresponding subset of paths, $\mathcal{R}'' : \mathcal{T}'' \rightsquigarrow_{24} E^Q$. For each $1 \leq i \leq k$, let $\mathcal{P}''_i = \mathcal{P}_i \cap \mathcal{P}''$, and let $\mathcal{M}''_i$ be the set of pairs of endpoints of the paths in $\mathcal{P}''_i$.

Let H be the graph given by Theorem 11. We are now ready to define fake demand pairs for the graph H. For each $1 \leq i \leq k$, we define a set $\tilde{\mathcal{M}}_i$ of demand pairs, and the sets $\tilde{S}_i, \tilde{T}_i$ of fake terminals, as follows: for each pair $(s,t) \in \mathcal{M}''_i$, if $Q_1 \in \mathcal{Q}$ is the critical cluster to which $f(s)$ belongs, and $Q_2 \in \mathcal{Q}$ the critical cluster to which $f(t)$ belongs, then we add (v_{Q_1}, v_{Q_2}) to $\tilde{\mathcal{M}}_i$, and we add v_{Q_1} to $\tilde{S}_i$ and v_{Q_2} to $\tilde{T}_i$. Notice that we allow $\tilde{\mathcal{M}}_i, \tilde{S}_i$ and $\tilde{T}_i$ to be multi-sets. This finishes the definition of the fake demand pairs. In order to complete Step 1, we show that any good integral solution to this new group-ICF instance will give a good integral solution to the original group-ICF instance. The proof of the next lemma appears in the full version of the paper.

Lemma 9 *Let $\tilde{\mathcal{P}}$ be any collection of paths in graph H that causes congestion at most γ. For each $1 \leq i \leq k$, let n_i be the number of paths in $\tilde{\mathcal{P}}$ connecting the fake terminals in $\tilde{S}_i$ to the fake terminals in $\tilde{T}_i$. Then we can efficiently find a collection $\mathcal{P}^*$ of paths in the original graph G, such that for each $1 \leq i \leq k$, there are n_i paths connecting the terminals of S_i to the terminals of T_i in $\mathcal{P}^*$, and the congestion caused by paths in $\mathcal{P}^*$ is at most $c_0\gamma$ for some constant c_0.*

Step 2: Routing in graph H.

In this step, we find a solution for the group-ICF problem defined on graph H and the set of fake terminal pairs. For each $1 \leq i \leq k$, let $\tilde{D}_i = |\tilde{\mathcal{M}}_i|$, and recall that $\tilde{D}_i = \Omega\left(\frac{D_i}{\log^{10} n}\right)$. For each $1 \leq i \leq k$, we will route a polylogarithmic fraction of the demand pairs in $\tilde{\mathcal{M}}_i$, with no congestion in graph H. This step is almost identical to the proof of Theorem 13, except that we use different parameters. For simplicity, we assume w.l.o.g. in this step that all values $\tilde{D}_i$ are equal. In order to achieve this, let D be the minimum value of $\tilde{D}_i$ over all $1 \leq i \leq k$. For each $1 \leq i \leq k$, we partition the demand pairs in $\tilde{\mathcal{M}}_i$ into subsets, containing D pairs each (except possibly the last subset). If one of the resulting subsets contains fewer than D pairs, we simply disregard all pairs in this subset. In this way, we define a new collection of demand pairs $\left\{\tilde{\mathcal{M}}'_{i'}\right\}$, where $1 \leq i' \leq k'$. Notice that it is now enough to find a collection $\tilde{\mathcal{P}}$ of paths, such that for each new group $\tilde{\mathcal{M}}'_{i'}$, at least a poly-logarithmic fraction of the pairs in $\tilde{\mathcal{M}}'_{i'}$ are connected by paths in $\tilde{\mathcal{P}}$. To simplify notation, we now assume w.l.o.g. that for each $1 \leq i \leq k$, $\tilde{D}_i = D$.

Let $\alpha_{\text{RZ}} = O(\log^{10} n)$, $L_{\text{RZ}} = \Theta(\log^5 n)$ be the parameters from Theorem 15. Let $x = 16m\alpha_{\text{RZ}} \cdot \log n = O(\log^{12} n)$. We set $L = 2\alpha^* \cdot x \cdot L_{\text{RZ}} = O(\log^{25} n \operatorname{poly}\log\log n)$.

We split graph H into x graphs $H_1, \ldots, H_x$, as follows. For each $1 \leq j \leq x$, we have $V(H_j) = V(H)$. In order to define the edge sets of graphs H_j, each edge $e \in E$, chooses an index $1 \leq j \leq x$ independently uniformly at random, and is then added to $E(H_j)$. This completes the definition of the graphs H_j.

For convenience, we define a new graph G', which is obtained from the original graph G by contracting each cluster $Q \in \mathcal{Q}$ into a super-node v_Q. Notice that the set $\tilde{\mathcal{M}}$ of fake terminal pairs is also a set of pairs of vertices in graph G', so we can also view $\tilde{\mathcal{M}}$ as defining a set of demand pairs in graph G'.

Given any partition (A, B) of the vertices of $V(H)$, let $\text{cut}_{G'}(A, B)$ denote the value of the minimum cut $|E_{G'}(A', B')|$ in graph G', such that $A \subseteq A', B \subseteq B'$. Theorem 11 guarantees that the size of the minimum cut in H is at least L/α^*, and for each partition (A, B) of $V(H)$, $\text{cut}_{G'}(A, B) \leq \alpha^* \cdot |E_H(A, B)|$. From Theorem 12, for each graph H_j, for $1 \leq j \leq x$, w.h.p. we have that: (i) The value of the minimum cut in H_j is at least $\frac{L}{2\alpha^* x} = L_{\text{RZ}}$; and (ii) For any partition (A, B) of $V(H_j)$, $|E_{H_j}(A, B)| \geq \frac{\text{cut}_{G'}(A,B)}{2x\alpha^*}$.

We need the following lemma, whose proof appears in the full version of the paper.

Lemma 10 *For each $1 \leq j \leq x$, there is a fractional solution to the instance $(H_j, \tilde{\mathcal{M}})$ of group-ICF, where each demand pair in $\tilde{\mathcal{M}}$ sends $\frac{1}{6mx\alpha^*\beta_{\text{FCG}}}$ flow units to each other with no congestion.*

In the rest of the algorithm, we apply the algorithm of Rao-Zhou to each of the graphs $H_1, \ldots, H_x$ in turn, together with some subset $\tilde{\mathcal{M}}^j \subseteq \tilde{\mathcal{M}}$ of the fake demand pairs. The output of the iteration is a collection $\tilde{\mathcal{P}}^j$ of edge-disjoint paths in graph H_j connecting some demand pairs in $\tilde{\mathcal{M}}^j$. We say that a pair $(\tilde{S}_i, \tilde{T}_i)$ is satisfied in iteration j, iff $\tilde{\mathcal{P}}^j$ contains at least $\frac{D}{48mx\alpha^*\beta_{\text{FCG}}\alpha_{\text{RZ}}}$ paths connecting demand pairs in $\tilde{\mathcal{M}}_i$.

We now fix some $1 \leq j \leq x$ and describe the execution of iteration j. Let $I \subseteq \{1, \ldots, k\}$ be the set of indices i, such that $(\tilde{S}_i, \tilde{T}_i)$ was not satisfied in iterations $1, \ldots, j-1$. Let $\tilde{\mathcal{M}}' = \bigcup_{i\in I} \tilde{\mathcal{M}}_i$. From Lemma 10, there is a fractional solution F in graph H_j, where every demand pair in $\tilde{\mathcal{M}}'$ sends $\frac{1}{6mx\alpha^*\beta_{\text{FCG}}}$ flow units to each other with no congestion. We transform instance $(H_j, \tilde{\mathcal{M}}')$ of group-ICF into a canonical instance, obtaining, for each $i \in I$, a collection $\tilde{\mathcal{P}}'_i$ of $\lfloor \frac{D}{6x\alpha^*\beta_{\text{FCG}}} \rfloor$ paths, such that the set $\bigcup_{i\in I} \tilde{\mathcal{P}}'_i$ of paths causes congestion at most $2m$ in graph H_j. Let $\tilde{\mathcal{M}}^j$ be the collection of pairs of endpoints of paths in $\bigcup_{i\in I} \tilde{\mathcal{P}}'_i$. We apply Theorem 15 to the EDP instance defined by the graph H_j and the set $\tilde{\mathcal{M}}^j$ of the demand pairs. Let $\tilde{\mathcal{P}}^j$ be the

output of the algorithm. Then w.h.p., $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2m\alpha_{\text{RZ}}}$, and the paths in $\tilde{\mathcal{P}}^j$ are edge-disjoint.

It is easy to verify that at least $\frac{1}{8m\alpha_{\text{RZ}}}$-fraction of pairs $(\tilde{S}_i, \tilde{T}_i)$ for $i \in I$ become satisfied in iteration j. This is since $|\tilde{\mathcal{P}}^j| \geq \frac{|\tilde{\mathcal{M}}^j|}{2m\alpha_{\text{RZ}}} \geq \frac{|I| \cdot D}{24m\alpha_{\text{RZ}} x\alpha^* \beta_{\text{FCG}}}$, each unsatisfied pair contributes at most $\frac{D}{48m\alpha_{\text{RZ}} x\alpha^* \beta_{\text{FCG}}}$ paths to $\tilde{\mathcal{P}}^j$, and each satisfied pair contributes at most $\frac{D}{6x\alpha^* \beta_{\text{FCG}}}$ paths. Therefore, after $x = 16m\alpha_{\text{RZ}} \cdot \log n$ iterations, all demand pairs are satisfied.

Let $\tilde{\mathcal{P}} = \bigcup_{j=1}^{x} \tilde{\mathcal{P}}^j$ denote the final collection of paths. For each demand pair $(\tilde{S}_i, \tilde{T}_i)$, set $\tilde{\mathcal{P}}$ must contain at least $\frac{\tilde{D}_i}{48mx\alpha^* \beta_{\text{FCG}} \cdot \alpha_{\text{RZ}}} = \Omega\left(\frac{D_i}{\log^{42} n \operatorname{poly}\log\log n}\right)$ paths connecting the vertices of $\tilde{S}_i$ to the vertices of $\tilde{T}_i$, and the paths in $\tilde{\mathcal{P}}$ are edge-disjoint. Applying Lemma 9, we obtain a collection $\mathcal{P}^*$ of paths in graph G, that cause a constant congestion, and for each $1 \leq i \leq k$, at least $\Omega\left(\frac{D_i}{\log^{42} n \operatorname{poly}\log\log n}\right)$ paths connect the vertices of S_i to the vertices of T_i. □

7.2 The Algorithm

We now present an algorithm to solve a general canonical instance. Our algorithm uses the following parameter:

$$\Delta = \max \begin{cases} 640kmL\alpha_{\text{ARV}}(n) \log n \\ \alpha_{\text{EDP}} \\ 16mL\alpha_{\mathsf{good}} c_{\mathsf{good}} \\ \Omega(L^2 \log^{11} n) \end{cases}$$

which gives $\Delta = k \operatorname{poly}\log n$. Let $\mathcal{T} = \bigcup_{i=1}^{k} (S_i \cup T_i)$ be the set of all terminals. The main idea is that we would like to find a Q-J decomposition of the graph G, such that each small cluster $X \in \mathcal{X}$, where $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{Q \in \mathcal{Q}} \pi(Q)\right)$, contains at most $D/\operatorname{poly}\log n$ terminals. Suppose we can find such a decomposition. We then say that a path $P \in \mathcal{P}$ is of type 1 iff its both endpoints are contained in some set $X \in \mathcal{X}$, and it is of type 2 otherwise. We can then partition the demand pairs (S_i, T_i) into two types: a demand pair (S_i, T_i) is of type 1 if most of the paths in $\mathcal{P}_i$ are of type 1, and it is of type 2 otherwise. This partitions the problem instance into two sub-instances: one induced by type-1 demand pairs, and the other induced by type-2 demand pairs. The former is a split instance w.r.t. $\mathcal{X}$, while for the latter instance, the current Q-J decomposition is a good decomposition. We can then solve the two resulting sub-instances using Theorems 19 and 20, respectively.

We are however unable to find such a Q-J decomposition directly. Instead, our algorithm consists of three steps. In the first step, we find a partition of $V(G)$ into subsets $V_1, \ldots, V_r$, and for each $1 \leq h \leq r$, we let $G_h = G[V_h]$. This partition guarantees that for each resulting graph G_h, for each small cut (A, B) in graph G_h, either A or B contain at most Δ terminals. Moreover, the number of edges $e \in E(G)$ whose endpoints do not lie in the same set V_h is at most $D/4$. This partition decomposes our original problem into r sub-problems, where for $1 \leq h \leq r$, the hth sub-problem is defined over the graph G_h. In the second step, for each graph G_h, we find a Q-J decomposition $(\mathcal{Q}_h, \mathcal{J}_h)$. Let $\mathcal{X}_h = \mathcal{J} \cup \{\pi(Q) \mid Q \in \mathcal{Q}_h\}$ be the corresponding set of small clusters. While the Q-J decomposition is not necessarily good, we ensure that each cluster $C \in \mathcal{X}_h$ may only contain a small number of pairs (s_j^i, t_j^i) for all $1 \leq i \leq k$. We then decompose the demand pairs (S_i, T_i) into several types, and define a separate sub-instance for each of these types. We will ensure that each one of the resulting instances is either a split instance, or the Q-J decomposition computed in step 2 is a good decomposition for it.

Step 1: Partitioning the graph G.

This step is summarized in the following lemma, whose proof is similar to the proof of standard well-linked decomposition and appears in the full version of the paper.

Lemma 11 *Let $(G, \mathcal{D})$ be a canonical instance of* group-ICF *with uniform demands. Then there is an efficient algorithm to find a partition $\mathcal{R}$ of V, such that for each $R \in \mathcal{R}$, for any partition (A, B) of R, where $|E_{G[R]}(A, B)| < 2L$, either A or B contain at most Δ terminals. Moreover, $\sum_{R \in \mathcal{R}} |\operatorname{out}(R)| \leq D/4$.*

We apply Lemma 11 to our instance $(G, \mathcal{D})$, to obtain a partition $\mathcal{R} = \{V_1, \ldots, V_r\}$ of $V(G)$. We denote $E' = \bigcup_{R \in \mathcal{R}} \operatorname{out}(R)$, and for each $1 \leq h \leq r$, we denote $G_h = G[V_h]$. Consider now some demand pair (S_i, T_i). Since $|E'| \leq D/4$, and the set $\mathcal{P}_i$ of paths causes congestion at most $2m$ in G, there are at least $mD/2$ pairs $(s_j^i, t_j^i) \in \mathcal{M}_i$, for which the path P_i^j is completely contained in some graph G_h. Let $\mathcal{M}_i' \subseteq \mathcal{M}_i$ be the subset of all such pairs (s_j^i, t_j^i), $|\mathcal{M}_i'| \geq mD/2$, and let $\mathcal{P}_i' \subseteq \mathcal{P}_i$ be the subset of their corresponding paths.

For each $1 \leq h \leq r$, let $\mathcal{M}_{i,h} \subseteq \mathcal{M}_i'$ be the subset of pairs (s_j^i, t_j^i) for which P_i^j is contained in G_h, let $\mathcal{P}_{i,h} \subseteq \mathcal{P}_i'$ be the subset of paths connecting such pairs, and let $D_{i,h} = |\mathcal{P}_{i,h}|$. Notice that $\sum_{h=1}^{r} |\mathcal{P}_{i,h}| \geq Dm/2$. For each $1 \leq h \leq r$, let $\mathcal{P}^h = \bigcup_{i=1}^{k} \mathcal{P}_{i,h}$, and let f^h be the fractional solution associated with the set $\mathcal{P}^h$ of paths, where each path in $\mathcal{P}^h$ is assigned $1/(2m)$ flow units. Then for each $1 \leq i \leq k$, flow f^h routes $D_{i,h}/(2m)$ flow units between the pairs in $\mathcal{M}_{i,h}$, and the flow f^h causes no congestion in G^h. We let $\mathcal{T}^h$ be the set of all terminals participating in pairs in $\bigcup_{i=1}^{k} \mathcal{M}_{i,h}$.

Step 2: constructing Q-J decompositions.

We say that a graph G_h is small iff $|\mathcal{T}^h| \leq 8\Delta$, and otherwise we say that it is large. The goal of this step is to find a Q-J decomposition for each large graph G_h. In this step we fix some graph $G' = G_h$, where G_h is a large graph, and we focus on finding a Q-J decomposition for it. We denote $\mathcal{T}' = \mathcal{T}^h$ to simplify notation. Recall that $|\mathcal{T}'| > 8\Delta$.

Suppose we are given a Q-J decomposition $(\mathcal{Q}, \mathcal{J})$ of G', and let $\mathcal{X} = \mathcal{J} \cup \left(\bigcup_{C \in \mathcal{Q}} \pi(C)\right)$. We say that this decomposition is *non-trivial* iff $\mathcal{Q} \neq \emptyset$. We say that it is *successful* iff each cluster $X \in \mathcal{X}$ contains at most Δ terminals in $\mathcal{T}'$. Notice that in general, Lemma 11 ensures that every cluster $X \in \mathcal{X}$ contains either at most Δ or at least $|\mathcal{T}'| - \Delta$ terminals from $\mathcal{T}'$, since for each $X \in \mathcal{X}$, $|\operatorname{out}_{G'}(X)| \leq L$. In order for a decomposition to be successful, we need to ensure that the latter case never happens.

We will instead achieve a decomposition with slightly weaker properties. Let $S^* \subseteq V(G')$ be any vertex subset containing at most Δ terminals, with $|\operatorname{out}_{G'}(S^*)| \leq 2L$ (we do not require that $G'[S^*]$ is connected). Let G_{S^*} be the graph obtained from G' as follows: if $|\operatorname{out}_{G'}(S^*)| \leq L$, then we set $G_{S^*} = G' \setminus S^*$. Otherwise, $L < |\operatorname{out}_{G'}(S^*)| \leq 2L$, and we obtain G_{S^*} from G' by contracting the vertices of S^* into a

super-node v_{S^*}. In this step, we show an efficient algorithm to find a set S^* with $|\operatorname{out}_{G'}(S^*)| \leq 2L$, together with a successful non-trivial Q-J decomposition for the graph G_{S^*}. The algorithm is summarized in the next theorem, whose proof appears in the full version of the paper.

Theorem 21 *There is an efficient algorithm to find a cluster $S^* \subseteq V(G')$ containing at most Δ terminals, with $|\operatorname{out}_{G'}(S^*)| \leq 2L$, and a successful non-trivial Q-J decomposition for the graph G_{S^*}.*

For each graph G_h, if G_h is large, we invoke Theorem 21 to obtain set S_h^* and $(\mathcal{Q}_h, \mathcal{J}_h)$, a Q-J decomposition of graph $G_{S_h^*}$. We denote $\mathcal{X}_h = \mathcal{J}_h \cup \left(\bigcup_{Q\in\mathcal{Q}_h} \pi(Q)\right)$, and $\mathcal{X}'_h = \mathcal{X}_h \cup \{S_h^*\}$. Otherwise, if G_h is small, then we denote $\mathcal{X}'_h = \{V(G_h)\}$. Finally, let $\mathcal{X} = \bigcup_{h=1}^{r} \mathcal{X}'_h$.

Consider some path $P \in \mathcal{P}'$. We say that this path is of type 1 iff it is completely contained in some cluster $X \in \mathcal{X}$. Assume now that the endpoints of P are contained in some cluster $X \in \mathcal{X}$, but P is not completely contained in cluster X. If $X = S_h^*$ for some $1 \leq h \leq r$, then we say that P is of type 2; otherwise, it is of type 3. All remaining paths are of type 4.

We partition the demand pairs (S_i, T_i) into four types. We say that a demand pair (S_i, T_i) is of type 1, iff at least 1/5 of the paths in $\mathcal{P}'_i$ are of type 1; we say that it is of type 2 iff at least 1/5 of the paths in $\mathcal{P}'_i$ are of type 2; similarly, it is of type 3 iff at least 1/5 of the paths in $\mathcal{P}'_i$ are of type 3, and otherwise it is of type 4. If a pair belongs to several types, we select one of the types for it arbitrarily.

Step 3: Routing the demands.

We route the demands of each one of the four types separately.

Type-1 demands.

It is easy to see that type-1 demands, together with the collection $\mathcal{X}$ of clusters, define a split instance. This is since each cluster $X \in \mathcal{X}$ contains at most Δ demand pairs, and $\Delta \leq \frac{D}{640m\alpha_{\text{EDP}}\cdot\ln^2 n}$ from Equation (1). If (S_i, T_i) is a type-1 demand, then the number of type-1 paths in $\mathcal{P}'_i$ is at least $Dm/10$. Therefore, we can apply Theorem 19, and obtain a collection $\mathcal{R}^1$ of paths that cause congestion at most η_{EDP} in G, and for each type-1 pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices in S_i to the vertices in T_i in R^1 w.h.p.

Type-2 Demands.

We show that the set of type-2 demands, together with the collection of vertex subsets V_h where G_h is large, define a valid split instance. Indeed, for each such subset V_h of vertices, every type-2 path that is contained in V_h must contain an edge in $\operatorname{out}_{G_h}(S_h^*)$. Since there are at most $2L$ such edges, and the paths in $\mathcal{P}$ cause congestion at most $2m$, we get that the number of type-2 paths contained in each such subset V_h is bounded by $4mL < \Delta \leq \frac{D}{640m\alpha_{\text{EDP}}\cdot\log^2 n}$. For each type-2 demand pair (S_i, T_i), there are at least $\frac{Dm}{10}$ type-2 paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{P}'_i$. Therefore, we can apply Theorem 19, and obtain a collection $\mathcal{R}^2$ of paths that cause congestion at most η_{EDP} in G, and for each type-2 demand pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices in S_i to the vertices in T_i in $\mathcal{R}^2$ w.h.p.

Type-3 Demands.

Let $X \in \mathcal{X} \setminus \{S_1^*, \ldots, S_r^*\}$, and consider the set $\mathcal{P}(X)$ of type-3 paths whose both endpoints belong to X. Assume w.l.o.g. that $X \subseteq V_h$. Since $|\operatorname{out}_{G_h}(X)| \leq 2L$, $|\mathcal{P}(X)| \leq 4Lm$ must hold. Recall that $G_h[X]$ is a connected graph if $X \notin \{S_1^*, \ldots, S_r^*\}$. Let $\mathcal{M}(X)$ be the collection of pairs of endpoints of the paths in $\mathcal{P}(X)$. We select one pair $(s,t) \in \mathcal{M}(X)$ uniformly at random, and we connect s to t by any path contained in $G[X]$. Let $\mathcal{R}^3$ be the set of all such resulting paths. Using the same arguments as in Theorem 19, it is easy to see that w.h.p. every type-3 demand pair (S_i, T_i) has at least $D/\operatorname{poly}\log n$ paths connecting the vertices of S_i to the vertices of T_i in $\mathcal{R}^3$, since $4mL < \frac{D}{16\log^2 n}$.

Type-4 Demands.

Let (S_i, T_i) be any type-4 demand, and let $\mathcal{P}_i^4 \subseteq \mathcal{P}'_i$ be the subset of type-4 paths for (S_i, T_i). Recall that $|\mathcal{P}_i^4| \geq Dm/5$. For each $1 \leq h \leq r$, let $\mathcal{P}_i^4(h) \subseteq \mathcal{P}_i^4$ be the subset of paths contained in the graph G_h. We say that pair (S_i, T_i) is *light* for G_h iff

$$|\mathcal{P}_i^4(h)| < \max\left\{16mL\alpha_{\text{good}}c_{\text{good}}, \Omega(L^2\log^{11} n)\right\}.$$

Otherwise, we say that it is *heavy* for G_h. We say that a demand pair (S_i, T_i) is light iff the total number of paths in sets $\mathcal{P}_i^4(h)$, where (S_i, T_i) is light for G_h is at least $Dm/10$. Otherwise, we say that it is heavy.

Let (S_i, T_i) be any demand pair, and let P be any type-4 path connecting a vertex of S_i to a vertex of T_i. Assume w.l.o.g. that P is contained in G_h for some $1 \leq h \leq r$. We say that P is a light path if (S_i, T_i) is light for G_h, and we say that it is a heavy path otherwise.

We now construct two canonical instances. The first instance consists of light (S_i, T_i) demand pairs of type 4, and their corresponding light type-4 paths in $\mathcal{P}'_i$. It is easy to see that this defines a split instance for the collection of vertex subsets V_h, where G_h is large. This is since for each light pair (S_i, T_i), for each subset V_h where (S_i, T_i) is light for G_h, $|\mathcal{P}_i^4(h)| < \max\left\{16mL\alpha_{\text{good}}c_{\text{good}}, \Omega(L^2\log^{11} n\right\} \leq \Delta \leq \frac{D}{640m\alpha_{\text{EDP}}\cdot\ln^2 n}$. Therefore, we can use Theorem 19 to find a collection $\mathcal{R}^4$ of paths that cause congestion at most η_{EDP}, and for each light type-4 pair (S_i, T_i), at least $D/\operatorname{poly}\log n$ paths connect the vertices of S_i to the vertices of T_i in $\mathcal{R}^4$ w.h.p.

Finally, consider some heavy type-4 pair (S_i, T_i). Let $\mathcal{P}''_i \subseteq \mathcal{P}'_i$ be the set of all heavy type-4 paths in $\mathcal{P}'_i$, and let $\mathcal{M}''_i$ be the set of pairs of their endpoints. Recall that $|\mathcal{M}''_i| \geq Dm/10$, and the paths in $\mathcal{P}''_i$ cause congestion at most $2m$ in G. For each $1 \leq h \leq r$, where G_h is large, let $\mathcal{P}''_i(h) \subseteq \mathcal{P}''_i$ be the subset of paths contained in G_h, and let $\mathcal{M}''_i(h)$ be the set of their endpoints.

Consider some large graph G_h, and consider the sets (S'_i, T'_i) of demands for $1 \leq i \leq k$, where S'_i contains the first endpoint and T'_i contains the last endpoint of every path in $\mathcal{P}''_i(h)$. Then the Q-J decomposition that we have computed in Step 2 is a good decomposition for graph $G'_{S_h^*}$, where $G' = G_h$, for the set $(S'_1, T'_1), \ldots, (S'_k, T'_k)$ of demands. Therefore, we can apply Theorem 20 to find a

collection $\mathcal{R}^5(h)$ of paths in graph $G'_{S_h^*}$ that cause congestion at most c_{good} in $G'_{S_h^*}$, and for each $1 \leq i \leq k$, at least $\lfloor \frac{|\mathcal{P}_i''(h)|}{2m\alpha_{\mathsf{good}}} \rfloor > 4Lc_{\mathsf{good}}$ paths connect vertices of S_i to vertices of T_i in $\mathcal{R}^5(h)$. Observe however that it is possible that $G'_{S_h^*}$ is obtained from G_h by contracting the vertices of S_h^* into a supernode v_h, and it is possible that some paths in $\mathcal{R}^5(h)$ contain the vertex v_h. However, since the degree of v_h is bounded by $2L$, and the congestion due to paths in $\mathcal{R}^5(h)$ is at most c_{good}, there are at most $2Lc_{\mathsf{good}}$ such paths in $\mathcal{R}^5(h)$. We simply remove all such paths from $\mathcal{R}^5(h)$. Since for each $1 \leq i \leq k'$, $\mathcal{R}^5(h)$ contains more than $4Lc_{\mathsf{good}}$ paths connecting S_i to T_i, we delete at most half the paths connecting each pair (S_i, T_i) in set $\mathcal{R}^5(h)$

Let $\mathcal{R}^5 = \bigcup_h \mathcal{R}^5(h)$. Then for each heavy type-4 pair (S_i, T_i), at least $\frac{Dm}{40m\alpha_{\mathsf{good}}} = \frac{D}{\text{poly}\log n}$ paths connect S_i to T_i in $\mathcal{R}^5$, since we are guaranteed that for all h, either $D_i(h) = 0$, or $D_i(h) \geq 2m\alpha_{\mathsf{good}}$. The congestion due to paths in $\mathcal{R}^5$ is bounded by c_{good}.

Our final solution is $\mathcal{P}^* = \bigcup_{j=1}^{5} \mathcal{R}^j$. From the above discussion, for every pair (S_i, T_i), set $\mathcal{P}^*$ contains at least $D/\text{poly}\log n$ paths connecting S_i to T_i, and the congestion due to $\mathcal{P}^*$ is bounded by a constant.

Acknowledgements.

The second author would like to thank Matthew Andrews for sharing an early version of his paper and for many interesting discussions. She also thanks Sanjeev Khanna for many interesting discussions about routing problems. The third author thanks Chandra Chekuri for useful discussions.

8. REFERENCES

[1] Matthew Andrews. Approximation algorithms for the edge-disjoint paths problem via Raecke decompositions. In *Proceedings of the 2010 IEEE 51st Annual Symposium on Foundations of Computer Science*, FOCS '10, pages 277–286, Washington, DC, USA, 2010. IEEE Computer Society.

[2] Matthew Andrews, Julia Chuzhoy, Venkatesan Guruswami, Sanjeev Khanna, Kunal Talwar, and Lisa Zhang. Inapproximability of edge-disjoint paths and low congestion routing on undirected graphs. *Combinatorica*, 30(5):485–520, 2010.

[3] Matthew Andrews and Lisa Zhang. Hardness of the undirected edge-disjoint paths problem. In Harold N. Gabow and Ronald Fagin, editors, *STOC*, pages 276–283. ACM, 2005.

[4] Matthew Andrews and Lisa Zhang. Hardness of the undirected congestion minimization problem. *SIAM J. Comput.*, 37(1):112–131, 2007.

[5] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[6] S. Arora and S. Safra. Probabilistic checking of proofs: a new characterization of NP. *Journal of the ACM*, 45(1):70–122, 1998.

[7] Sanjeev Arora, Satish Rao, and Umesh V. Vazirani. Expander flows, geometric embeddings and graph partitioning. *J. ACM*, 56(2), 2009.

[8] Yonatan Aumann and Yuval Rabani. An $O(\log k)$ approximate min-cut max-flow theorem and approximation algorithm. *SIAM J. Comput.*, 27(1):291–301, 1998.

[9] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. The all-or-nothing multicommodity flow problem. In *Proceedings of the thirty-sixth annual ACM symposium on Theory of computing*, STOC '04, pages 156–165, New York, NY, USA, 2004. ACM.

[10] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. Multicommodity flow, well-linked terminals, and routing problems. In *STOC '05: Proceedings of the thirty-seventh annual ACM symposium on Theory of computing*, pages 183–192, New York, NY, USA, 2005. ACM.

[11] Chandra Chekuri, Sanjeev Khanna, and F. Bruce Shepherd. An $O(\sqrt{n})$ approximation and integrality gap for disjoint paths and unsplittable flow. *Theory of Computing*, 2(1):137–146, 2006.

[12] Julia Chuzhoy. Routing in undirected graphs with constant congestion. In *STOC 2012, to appear*.

[13] Julia Chuzhoy and Joseph (Seffi) Naor. New hardness results for congestion minimization and machine scheduling. *J. ACM*, 53(5):707–721, 2006.

[14] M. Conforti, R. Hassin, and R. Ravi. Reconstructing flow paths. *Operations Research Letters*, 31:273–276, 2003.

[15] David Gale and Lloyd Shapley. College admissions and the stability of marriage. *American Mathematical Monthly*, 1:9–14, 1962.

[16] N. Garg, V.V. Vazirani, and M. Yannakakis. Approximate max-flow min-(multi)-cut theorems and their applications. *SIAM Journal on Computing*, 25:235–251, 1995.

[17] Naveen Garg, Vijay V. Vazirani, and Mihalis Yannakakis. Primal-dual approximation algorithms for integral flow and multicut in trees, with applications to matching and set cover. In Andrzej Lingas, Rolf G. Karlsson, and Svante Carlsson, editors, *ICALP*, volume 700 of *Lecture Notes in Computer Science*, pages 64–75. Springer, 1993.

[18] David R. Karger. Random sampling in cut, flow, and network design problems. *Mathematics of Operations Research*, 24:383–413, 1999.

[19] R. Karp. Reducibility among combinatorial problems. In R. Miller and J. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[20] F. T. Leighton and S. Rao. Multicommodity max-flow min-cut theorems and their use in designing approximation algorithms. *Journal of the ACM*, 46:787–832, 1999.

[21] N. Linial, E. London, and Y. Rabinovich. The geometry of graphs and some of its algorithmic applications. *Proceedings of 35th Annual IEEE Symposium on Foundations of Computer Science*, pages 577–591, 1994.

[22] Harald Räcke. Minimizing congestion in general networks. In *In Proceedings of the 43rd IEEE Symposium on Foundations of Computer Science (FOCS)*, pages 43–52, 2002.

[23] Prabhakar Raghavan and Clark D. Tompson. Randomized rounding: a technique for provably good

algorithms and algorithmic proofs. *Combinatorica*, 7:365–374, December 1987.

[24] Satish Rao and Shuheng Zhou. Edge disjoint paths in moderately connected graphs. *SIAM J. Comput.*, 39(5):1856–1887, 2010.

[25] R. Raz. A parallel repetition theorem. *SIAM J. Comput.*, 27(3):763–803, 1998.

[26] N. Robertson and P. D. Seymour. Outline of a disjoint paths algorithm. In *Paths, Flows and VLSI-Layout*. Springer-Verlag, 1990.

APPENDIX

A. ARBITRARY EDGE CAPACITIES AND DEMANDS

In this section we extend our algorithms for basic-ICF and for group-ICF from Sections 4 and 7 to arbitrary demands and edge capacities. We only present here the generalization for basic-ICF, since the extension of the algorithm for group-ICF to general edge capacities and demands is almost identical.

Let $\alpha = \text{poly}\log n$ denote the approximation factor of the algorithm from Section 4, and let γ denote the congestion. Recall that for each demand pair $(t,t') \in \mathcal{M}$, the algorithm finds $\lfloor \lambda_{\text{OPT}} D/\alpha \rfloor$ paths connecting t to t' in G.

We now assume that we are given a set $\mathcal{D}$ of arbitrary demands, and the edge capacities are also arbitrary. We assume w.l.o.g. that $\lambda_{\text{OPT}} = 1$. Let $D_{\max}$ and $D_{\min}$ be the maximum and the minimum demands in $\mathcal{D}$, respectively. We first consider the case where $D_{\max}/D_{\min} \leq n^3$, and show a factor 4α-approximation with congestion at most γ for it.

If $D_{\min} \geq 2n^3$, then we delete all edges whose capacities are less than $\rho = \lfloor D_{\min}/(2n^3) \rfloor$. Notice that the total amount of flow going through such edges in the optimal fractional solution is bounded by $D_{\min}/2$, so this change reduces the value of the optimal fractional solution by at most factor 2. We then divide all demands and edge capacities by the factor ρ, thus obtaining a new problem instance G'. The value of the optimal solution for G' remains $\lambda_{\text{OPT}} = 1$, and any integral solution of value λ and congestion η in G' can be converted into an integral solution of value λ and congestion η in G. From now on we will assume w.l.o.g. that the value of the minimum demand $D_{\min} \leq 4n^3$, and so the value of the maximum demand, $D_{\max} \leq 4n^6$, while $\lambda_{\text{OPT}} = 1$.

Let $D^* = 4\alpha/\lambda_{\text{OPT}}$. Since we are only interested in finding a factor 4α-approximation, we can assume w.l.o.g., that for each pair of terminals, either $D(t,t') = 0$, or $D(t,t') \geq D^*$. In particular, $D_{\min} \geq D^*$.

We now slightly modify the graph G, and define a new set $\mathcal{D}'$ of demands, such that in the new instance all capacities are unit, and the demands are uniform. Let $D = D_{\min}$. We start with $\mathcal{M} = \emptyset$. Consider some demand pair (s,t) with $D(s,t) > 0$, and let $N(s,t) = \lfloor D(s,t)/D \rfloor$. We create $N(s,t)$ copies of the source s that connect to s with a capacity-∞ edge each, and $N(s,t)$ copies of the sink t that connect to t with capacity-∞ edges. We also add $N(s,t)$ disjoint pairs of vertices, each containing one copy of s and one copy of t, to set $\mathcal{M}$. Let $\mathcal{M}$ be the final set of terminal pairs, obtained after we process all pairs with non-zero demands, and let $\mathcal{D}'$ be the corresponding set of demands, where for each pair $(s',t') \in \mathcal{M}$, we set its demand $D'(s',t')$ to be D. Notice that so far for each pair (s,t) of vertices with $D(s,t) > 0$, the total demand in $\mathcal{D}'$ between the copies of s and the copies of t is at least $D(s,t)/2$ and at most $D(s,t)$. Therefore, an α'-approximate solution to the resulting instance will give a $2\alpha'$-approximation to the original instance. Our final step is to take care of the non-uniform edge capacities. Since we are interested in finding an integral routing, we can assume w.l.o.g. that all edge capacities $c_e \geq 1$.

Since $\lambda_{\text{OPT}} = 1$, the total flow through any edge in the optimal fractional solution cannot exceed $n^2 \cdot D_{\max}$, so if the capacity of any edge is greater than $n^2 \cdot D_{\max}$, we can set it to $n^2 \cdot D_{\max}$ without changing the value of the optimal fractional solution. Finally, for each edge e, we replace e with $\lceil c(e) \rceil$ parallel edges with unit capacities. The resulting instance of ICF has unit edge capacities and uniform demands. The value of the optimal fractional solution is at least $\lambda_{\text{OPT}}/2$, where λ_{OPT} is the value of the optimal fractional solution in the original instance. We can now use the algorithm from Section 4 to find an α-approximate integral solution with congestion at most γ for this new instance. This solution immediately gives a factor 4α-approximation with congestion at most 2γ for the original instance.

Assume now that we are given an instance $(G,\mathcal{D})$ of basic-ICF with arbitrary demands and capacities. By appropriately scaling the demands, we can assume w.l.o.g. that $\lambda_{\text{OPT}} = 1$. We group the demands geometrically into groups $\mathcal{D}_1, \mathcal{D}_2, \ldots$, where group $\mathcal{D}_i$ contains all demands $D(s,t)$ with $n^{3(i-1)} D'_{\min} \leq D(s,t) < n^{3i} D'_{\min}$, where $D'_{\min}$ is the minimum demand in $\mathcal{D}$. Notice that the number of non-empty groups $\mathcal{D}_i$ is bounded by n^2. For each non-empty group $\mathcal{D}_i$, we create a new instance of basic-ICF, as follows. We build a graph G_i whose set of vertices is $V(G)$, and the set of edges consists of all edges of G whose capacities are at least $n^{3(i-2)} D'_{\min}$. If the capacity of an edge is more than $n^{3i+2} D'_{\min}$, then we set its capacity to n^{3i+2}. The capacities of all other edges remain unchanged. We then use the algorithm for the special case where $D_{\max}/D_{\min} \leq n^3$ for each one of the resulting instances $(G_i, \mathcal{D}_i)$, and output the union of their solutions. Since the value of the optimal fractional solution in each such instance is at least $\lambda_{\text{OPT}}/2$, it is immediate to verify that we obtain a factor 8α-approximation. In order to bound the edge congestion, observe that for each edge $e \in E(G)$, the total capacity of copies of edge e in all instances $(G_i, \mathcal{D}_i)$ to which e belongs is bounded by $4c(e)$. Therefore, the overall edge congestion is bounded by 8γ.